



# *wxWidgets*

## 初学者导引

迂者-贺利坚 编写

为 IT 菜鸟起飞铺跑道

<http://blog.csdn.net/sxhelijian>

2014 年 5 月

# 目录

1 前言.....	1
1.1 关于“导引” .....	1
1.2 为什么 wxWidgets.....	1
1.3 看本文的方法.....	2
1.4 看本文需要的基础.....	2
1.5 本文特点.....	3
2 下载、安装 wxWidgets.....	4
2.1 下载 wxWidgets.....	4
2.2 为什么要自己编译 wxWidgets.....	5
2.3 编译 wxWidgets 前的准备.....	6
2.4 编译 wxWidgets.....	7
2.4.1 用命令行编译 wxWidgets.....	7
2.4.2 意外处理.....	8
2.4.3 多知道一点.....	9
3 wxWidgets 应用程序初体验.....	10
3.1 由“空项目”建立和运行 GUI 应用程序.....	11
3.1.1 建立项目.....	11
3.1.2 编译和运行项目.....	15
3.2 利用 Code::Blocks 的向导建立应用.....	17
4 wxWidgets 学习资料及利用方法指导.....	23
4.1 关于 C++ wxWidgets 的书籍.....	23
4.1.1 《使用 wxWidgets 进行跨平台程序开发》 .....	23
4.1.2 wxwidgets 的 Wiki 主页.....	23
4.1.3 《wxWidgets tutorial》 .....	23
4.2 用好 wxWidgets 的在线文档.....	24
4.2.1 成熟平台常有在线文档.....	24
4.2.2 wxWidgets 的在线文档.....	25

4.2.3 查找在线文档.....	26
4.2.4 查看在线文档的设备支持.....	28
4.3 在编程环境中找帮助.....	28
4.4 深入学习路线建议.....	30
4.4.1 看书的策略.....	30
4.4.2 更多的案例.....	31
5 用 wxSmith 进行可视化设计.....	33
5.1 用 wxSmith 创建应用程序的外观.....	33
5.2 为控件加入事件处理程序.....	35
5.3 写代码与拖控件.....	38
5.4 深入学习的建议.....	38
附：学习材料清单.....	40

# 1 前言

## 1.1 关于“导引”

大学中的“C++程序设计”课程，以掌握基本的 C++语法，并运用其解决一般的计算问题为目的。学生在学习编出的程序，在“长相”上，与实际的产品不太一样。我的学生感慨，为何我编程序总是要面对“黑框框”，而日常使用的程序，那界面很友好。

对了，我说的是“程序和用户的接口”，俗称“界面”。

大学生在成长为一名合格的工程技术人员的过程中，需要多阶段、多环节的培养，一门课程抓住要解决的主要问题，其他课程再解决其他问题。大学的课程有这种阶段性的特点。在实际的工程中，用 C++做的“产品”，很多根本不需要界面，访问底层的代码、对性能要求高的关键计算，非 C/C++不可。界面，某种角度讲，不是核心。

但是，作为大众产品，没有好长相，面临的可能就是失败。无论如何，学习了 C++，要用 C++做出界面友好的程序，这个想法不能丢。作为学过 C++的同学，能做出一个漂亮的应用程序，例如俄罗斯方块游戏，那是一件很荣耀的事。这也可以作为一个新的阶段的开始。在大学，有了修过的课程做基础，这件事情可以在课外自行拓展。

为有 C++基础的学生，需要制定一个初步“进阶”的方案。本文就是要做这样的事。

## 1.2 为什么 wxWidgets

wxWidgets 是一个开源的跨平台的 C++构架库（framework），它可以提供 GUI（图形用户界面）和其它工具。wxWidgets 除了可以用于开发“有界面”的程序，还提供对图形、多媒体、网络等常见领域应用的支持。

掌握了 wxWidgets，就有办法搞定常见的需要让计算机完成的任务。有了使用 wxWidgets 开发的体验，也打开了引入其他构架开发的大门。

wxWidgets 是开源的，无论对于个人还是对于商业应用都是免费的。wxWidgets 可以支持现今几乎所有操作系统，包括对掌上电脑的支持。wxWidgets 社区快速稳健成长，其周边工具也越来越多。wxWidgets 支持各种主流的编译器，通过“重新编译”的方式支持软件的

移植。wxWidgets 吸引我的还有，尽可能的使用目标系统“原生的”的 GUI 样式，界面与环境异常和谐。

关于 wxWidgets 的好，上网搜索可以获得更多。

同样称为 C++ 构架库的，还有 MFC 和 Qt。

MFC 是微软制造的经典。当然，MFC 只适合 Windows 平台，也显老旧。对于这个时期刚起步的大学生而言，直接学习面向跨平台的开发，理所当然。能将我的学生引向读开源代码，也是我想努力的方向。

Qt 同样跨平台，同样开源。Qt 由商业机构维护，有人甚至说更出众。

哪种语言好，哪个平台强？为此打嘴仗的很多。其实对于初学者而言，重在过程，重在体验。早已经不是“从一而终”的时代了，何况身处 IT 这样一个变化快的行业中。用学习 A 的体验，具备学习 B 和 C 的能力，这是最重要的。

也许本文选 wxWidgets 都是一个偶然。借助 wxWidgets，获得 C++ 应用程序开发的初步体验，足矣。再有回报，都是额外的馈赠。

在本文中，还用了 Code::Blocks，一个优秀的开源 C++ IDE。为什么是 Code::Blocks，而不是 VS20xx，或者其他？同样无聊的问题，不去对比。实际情况是，我在教学中，我的学生会用多种 IDE，主要用 Code::Blocks。多加一句，Code::Blocks 还是很好的。

## 1.3 看本文的方法

只看不练空把式。本文，以及后续指出的学习路线，不是看下去的，而是做下去的。

光看不练，是白看。看完了，没有做，结果必然还是不会。看、练结合，要获得能实践的知识和技能。

边看边做，此为道。

## 1.4 看本文需要的基础

本文面向的是初学者，尤其是只按着教学要求学过了 C++ 相关课程，或者自学了一段时间 C++ 的大学生或程序设计爱好者。

- 程序设计的一般知识和技能：简单而言，学过 C++ 课程，做过了一些练习。掌握了 C++ 中类、对象、构造函数、重载、继承、多态等基本概念。要补课，可以到我的《C++ 程

序设计课程主页-2013 级》，<http://blog.csdn.net/sxhelijian/article/details/11890759> 中完成实践。

- 会用 Code::Blocks 调试程序: 如果以前是在其他平台中实践 C++ 程序设计, 请自行下载、安装 Code::Blocks, 然后编译、运行几个程序即可具备。Code::Blocks 的使用请看 <http://blog.csdn.net/sxhelijian/article/details/17710041>。

## 1.5 本文特点

本文最大的特点是虎头蛇尾。

这是作者的自我表扬, 不是批评。本文想要起到的作用无非两点:

### (1) 引导读者能够搭建起一个能开始工作的环境

配置环境的这汪水本来不浅, 这样的锻炼足够珍贵。但搭建环境的过程中, 有不少细节并不是一时半会儿就可以学会的, 不少内容有待后续的学习中解决和领悟。本文始终记着要用 wxWidgets 尽快做出程序来这样一个目标, 尽快地让读者能开始工作, 这是第一要务。

为此, 只要能将这个过程顺下去, 忽略中间需要知道的一切。在读者经历过后, 可以再走一遍, 会发现处处能有感悟。

### (2) 引导读者正确运用文档

在软件开发过程中, 用的最多的是在线文档, 而不是书本 (或许有些时候会将在线文档印成书)。能尽早学会使用在线文档中的链接, 找到需要的帮助信息, 对初学者的意义, 就是一场学习的革命。

之所以做虎头, 是因为万事开头难。本文只解决开头难的事情。开了头, 就好了。

之能以能这样做, 是因为, 只要开头, 就有无数多的资料为你所用。有人说 wxWidgets 的参考资料少, 那是因为他没有用好已有的资料。

先造个虎头, 让初学者自己续上虎身、虎尾。

## 2 下载、安装 wxWidgets

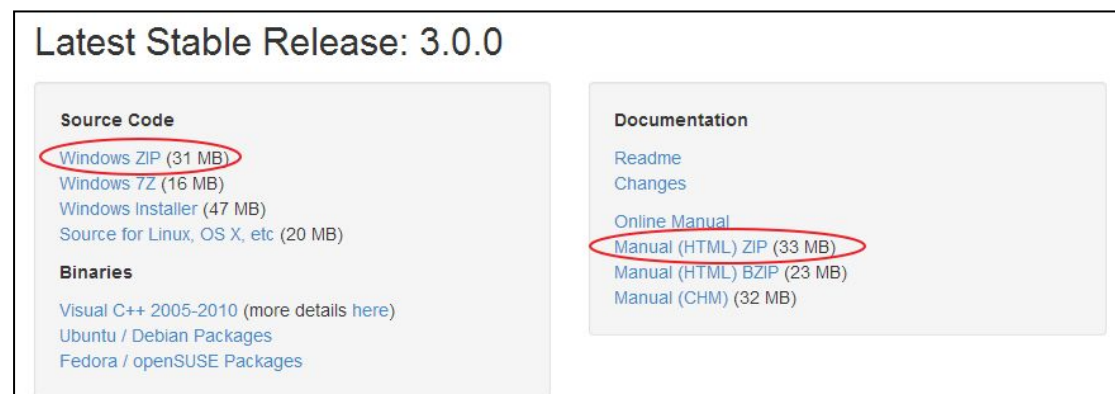
这方面的资料从网络可以找到不少。wxWidgets 的文档，要涵盖各种操作系统和编译环境，难免让人有点无所适从。这对初学者，是灾难。

以下文字，适合于大部分初学者的工作环境。为能边看边做，请确认：（1）你用的是 Windows 操作系统（强烈建议初学者进阶后，适时开启 Linux 下开发的体验）；（2）已经安装了 Code::Blocks（版本不限，但也别太低了）；（3）Code::Blocks 使用 gcc 编译器，随 Code::Blocks 的安装已经装好。

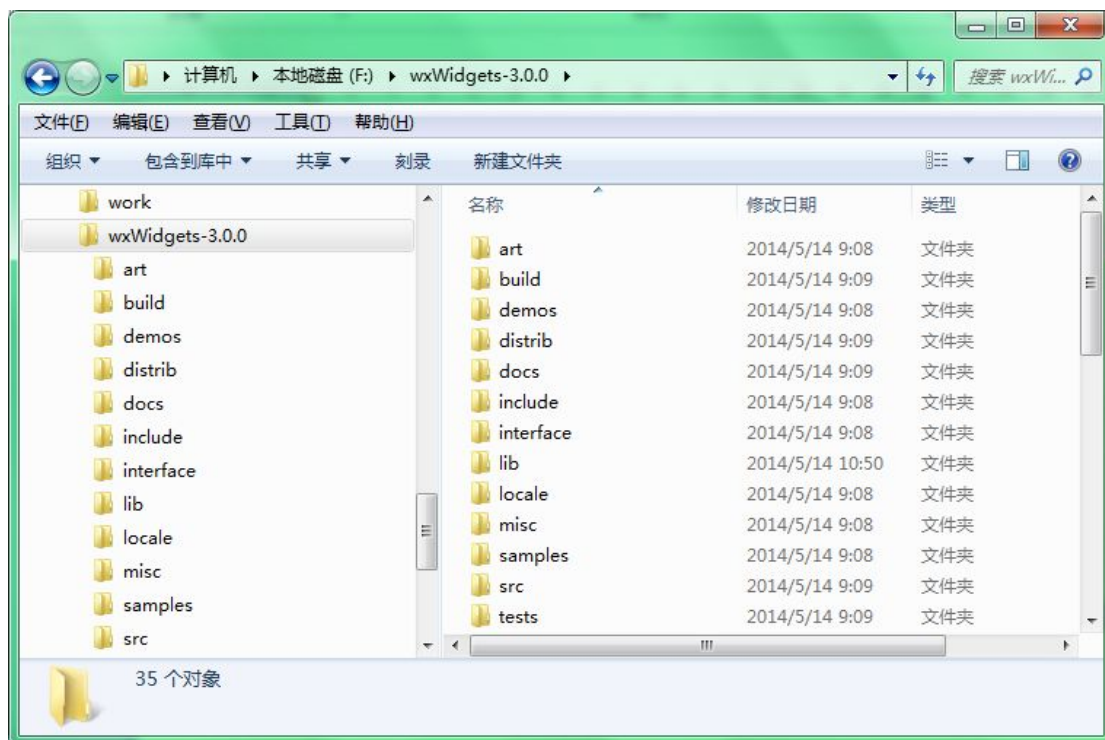
### 2.1 下载 wxWidgets

wxWidgets 的官网在 <http://www.wxwidgets.org/downloads/>，下载页面是 <http://www.wxwidgets.org/downloads/>。

作为 Windows 用户，从下载页面下载如下画圈的两个文件。Windows.ZIP 是 wxWidgets 的源代码，Manual(HTML).zip 则是在学习过程中最重要的参考（即前述的在线文档）。在 4.2 节中，将专门介绍在线文档的使用。



下载后，将文件 Windows.ZIP 解压缩，下图是我解压缩后的结果，我将其解压到了 F:/wxWidgets-3.0.0 中。后文中，我将用 X:/wxWidgets-3.0.0 表示这个文件夹，X 代表你选择的盘符。



## 2.2 为什么要自己编译 wxWidgets

一般的 Windows 应用程序，总是有一个安装程序（常常是 `setup.exe`），只要运行这个程序，就可以将软件安装好。

wxWidgets 不是这样。下载得到的，不是能运行的程序，而是 wxWidgets 的源代码！开源软件提供给用户源代码，你可以直接阅读和修改。

不少开源软件也提供安装程序，用户安装后就可以使用。这是适用于软件的使用者的方式，而不是针对开发者的方式。

现在，你是开发者。作为开发者，常是下载源代码后，自己编译。这对于大众是高要求，但对专业人员，却是常用的套路。

wxWidgets 不是一般的应用程序，是为支持应用程序开发的平台。wxWidgets 面对的是在不同操作系统（Linux、unix、Windows、Mac OS）下工作的开发人员，他们使用的 C++ 编译器（GCC 家族、MS 家族、Borland 家族及其他各种）形形色色、版本各异。wxWidgets 不便于提供各种组合下的安装程序。开发人员下载源码，自己编译自己用。这种方式，创建的是最适合自己的环境。

实际上，不这样做，往往得不到适合自己的开发环境。



所以，下面的步骤或许会有点挑战性。但不要有牢骚。这样做一遍，你作为开发者的成色，就更足一些。

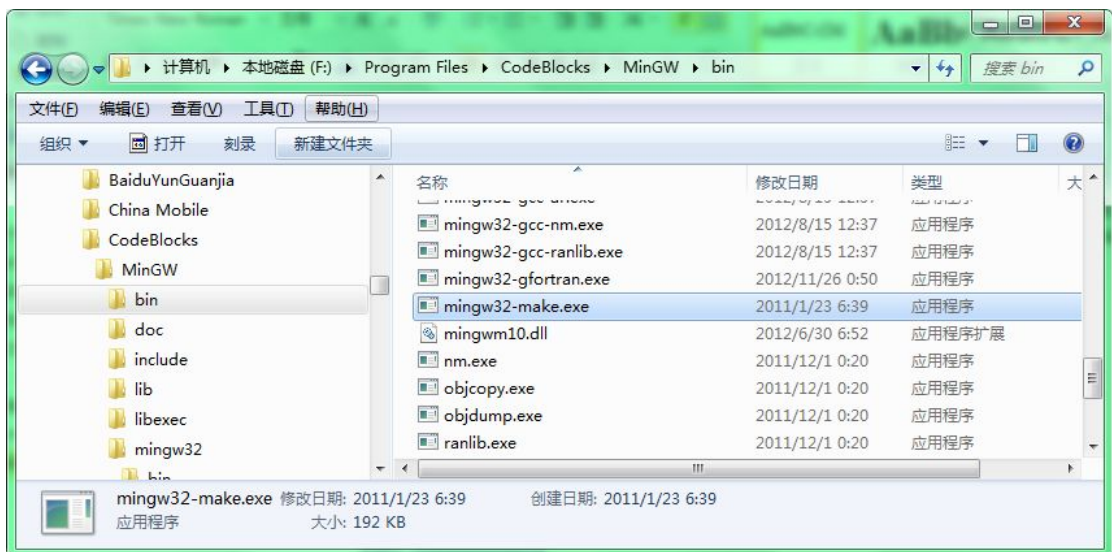
实际上，wxWidgets 中还提供了一个称为 wxPack 的编译好的版本，可以用于直接安装。在我的体验中，最新的 wxPack 使用的 GCC 版本低了，安装顺利，但却不能正确运行在我的开发环境中写的程序。这种方法，不推荐使用。

## 2.3 编译 wxWidgets 前的准备

编译 wxWidgets，要先准备好编译器，并且配置好运行编译器的“环境”。

对于初学者，安装 Code::Blocks 时，选择带 GCC 编译器的安装文件进行安装。单独安装的 GCC 编译器，也可以在 Code::Blocks 中通过设置进行工作。

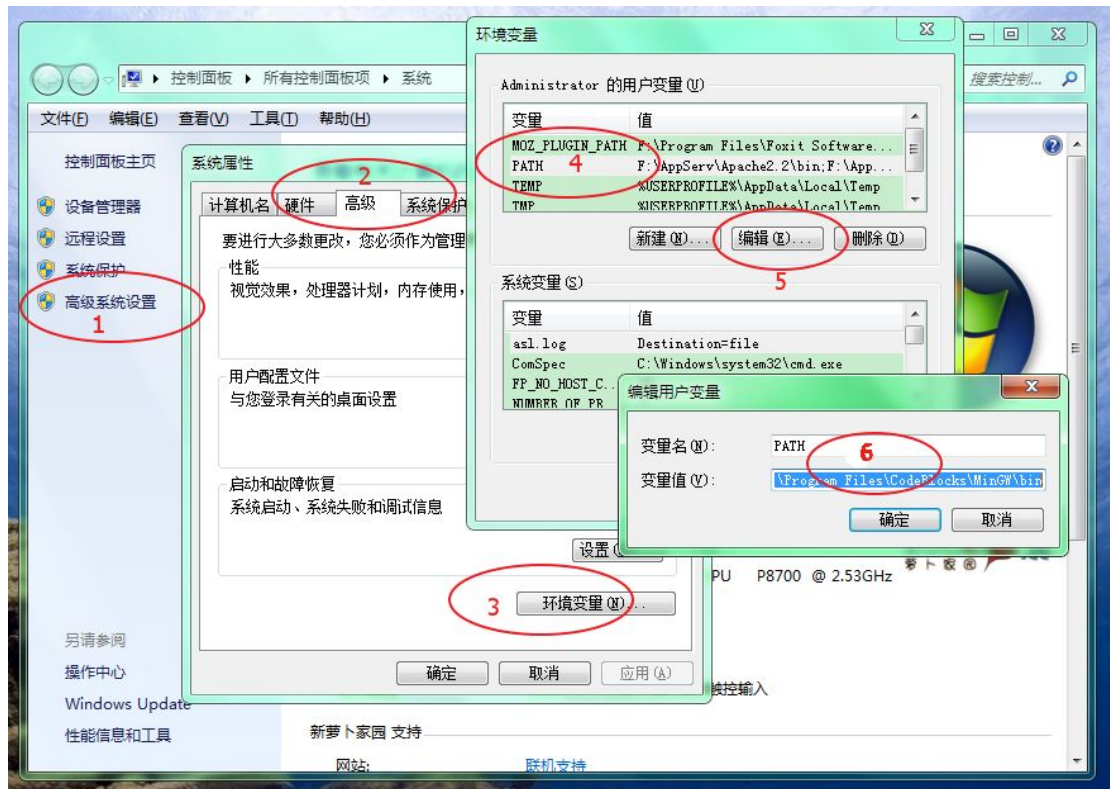
总之先找到 Code::Blocks 所在的文件夹(后文中我用“X:\CodeBlocks\”代表，X:\CodeBlocks\要替换为你使用的实际文件夹名)。随 Code::Blocks 安装的 GCC，在 Code::Blocks 安装文件夹的 MinGW 子文件夹中，打开 X:\CodeBlocks\MinGW\bin，其中的文件，如下图所示，就是支持你的 C++工作的 GCC 系列程序。



记下这个文件夹名（简单办法，将路径复制下来，暂时粘贴到一个文本文件中备用）。

下面要配置运行编译器的“环境”，确切地说，只需要设置“路径”（PATH）即可。

在 Win7 中，鼠标右击桌面上的“计算机”图标，选菜单中的“属性”，接下来，就是如下图中从 1 到 6 的一系列操作，将“变量名”为 PATH 的“变量值”，在原有值的后面加一个英文的分号，再加入你记下的 X:\CodeBlocks\MinGW\bin。注意，不要将原有的内容替换掉，而是追加你需要的路径即可。



其他版本的 Windows，找到“系统属性”对话框的方式可能会稍有不同，最终的目标都是设置好 PATH 的值。

还有别一种方式，直接用 DOS 命令做。有不少资料中讲这种做法，本文不做介绍。

## 2.4 编译 wxWidgets

编译 wxWidgets 的事情需要用命令行的方式完成。

### 2.4.1 用命令行编译 wxWidgets

从“开始”菜单->附件，运行“命令提示符”（有的系统称“MS-DOS 方式”，在命令行下分别输入下面的命令：

命令	解释
X:(回车)	当前盘置为 X，X 是你解压缩 wxWidgets 用的盘符。
cd \wxWidgets-3.0.0\build\msw(回车)	当前目录置为/wxWidgets-3.0.0\build\msw，可以查看这个文件夹中的文件，msw 是专供微软(ms)的 Windows(w)用的编译需要的文件（注：wxWidgets 的 C++源代码在\wxWidgets-3.0.0\src 中）。

gcc -v(回车)	这个命令并非必须，意在检查刚才的路径设置是否正确。下图的输出，表明在 X:\wxWidgets-3.0.0\build\msw 目录中，可以运行 X:\CodeBlocks\MinGW\bin 中的命令。还可以看到，当前使用的 GCC 版本是 4.7.1。
------------	--

下图是我运行上表中的命令出现的结果：

```

C:\Users\Administrator>F:
F:\>cd \wxWidgets-3.0.0\build\msw
F:\wxWidgets-3.0.0\build\msw>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=f:/program files/codeblocks/mingw/bin/./libexec/gcc/mingw32/4.7.1/lto-wrapper.exe
Target: mingw32
Configured with: ../../src/gcc-4.7.1/configure --build=mingw32 --enable-languages=c,c++,ada,fortran,objc,obj-c++ --enable-threads=win32 --enable-libgomp --enable-lto --enable-fully-dynamic-string --enable-libstdcxx-debug --enable-version-specific-runtime-libs --with-gnu-ld --disable-nls --disable-win32-registry --disable-symvers --disable-build-poststage1-with-cxx --disable-werror --prefix=/mingw32tdm --with-local-prefix=/mingw32tdm --enable-cxx-flags='-fno-function-sections -fno-data-sections' --with-pkgversion=tdm-1 --enable-sjlj-exceptions --with-bugurl=http://tdm-gcc.tdragon.net/bugs
Thread model: win32
gcc version 4.7.1 (tdm-1)
F:\wxWidgets-3.0.0\build\msw>
半:

```

下面就可以要开始编译 wxWidgets 了。就在 DOS 提示符后面，输入下面的命令：

```
mingw32-make -f makefile.gcc MONOLITHIC=1 SHARED=1 UNICODE=1 BUILD=debug
```

编译的过程会比较慢，会有几十分钟。干点别的，或者就看着屏幕上看不懂的提示发呆也好。理想情况是，顺利完成编译。

## 2.4.2 意外处理

我在编译 wxWidgets 中，苦等几十分钟，等来了一个 error，最后两行提示是：

```
gcc_mswuddll\monodll_xh_bmpcbox.o: file not recognized: Memory exhausted
collect2.exe: error: ld returned 1 exit status
```

出现这种情况的，到 <http://blog.csdn.net/sxhelijian/article/details/25749505> 中的“问题 1”，看原因解释以及对策。

### 2.4.3 多知道一点

用上面的命令编译后，可以满足学习的需求了。如果还想体验，以及支持将来生产用于发布的程序版本，可以在 SHARED 和 BUILD 参数的选取上再做些组合。

SHARED 的取值可以是 1 或 0，代表产生的是动态链接库（1）和静态链接库（0）。两者的区别不解释，以后将明白，或者自行百度之。

BUILD 的取值可以是 debug 或 release，代表在应用程序开发时，产生的可执行文件是调试版本（debug）还是发布版本（release）。

所以可以运行的命令还有 3 个：

```
mingw32-make -f makefile.gcc MONOLITHIC=1 SHARED=0 UNICODE=1 BUILD=debug
mingw32-make -f makefile.gcc MONOLITHIC=1 SHARED=1 UNICODE=1 BUILD=release
mingw32-make -f makefile.gcc MONOLITHIC=1 SHARED=0 UNICODE=1 BUILD=release
```

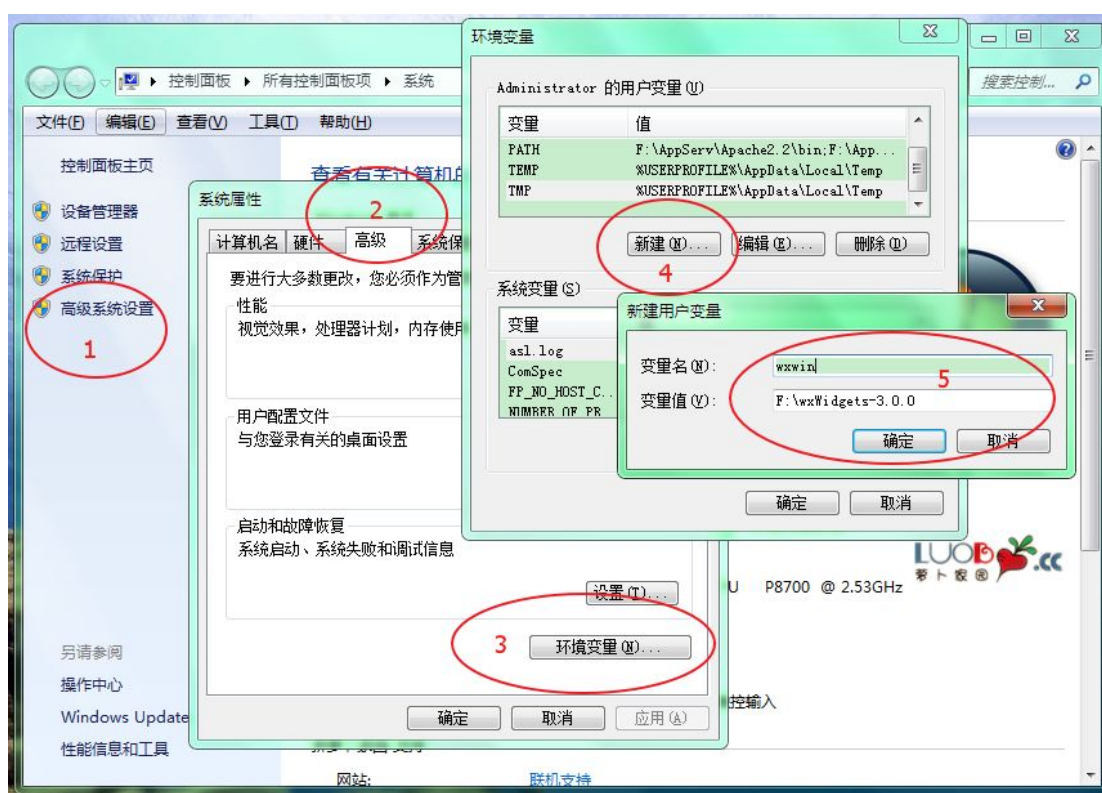
关于这些参数的解释，参考 <http://blog.csdn.net/sxhelijian/article/details/25749505> 中的“问题 2”部分。

### 3 wxWidgets 应用程序初体验

本文中所有的体验，在 Code::Blocks 中进行。

为了在 Code::Blocks 中编译运行 C++写的 wxWidgets 程序，需要再做些设置。

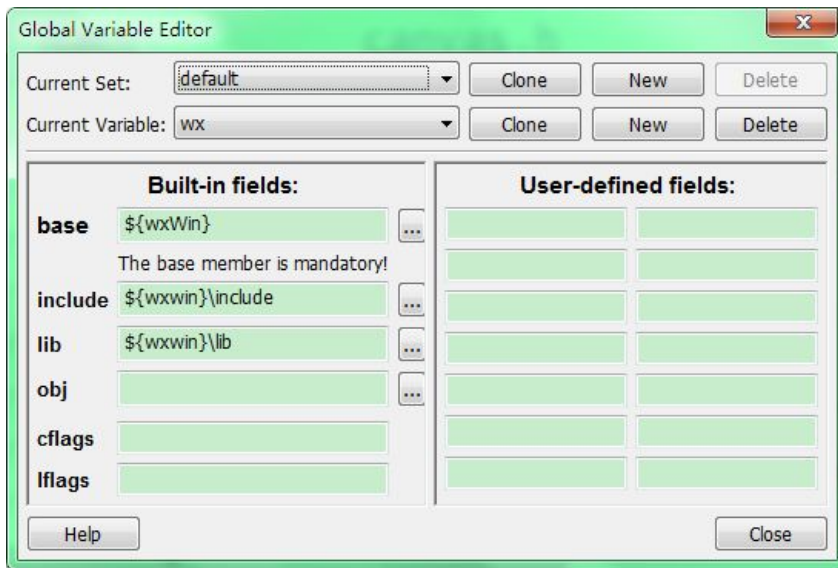
首先，需要在环境变量里添加一个 wxWidgets 根目录环境变量。设置方法类似于 2.3 中设置 PATH 变量的方法。在 Win7 中，右击桌面上的“计算机”图标，选菜单中的“属性”，在“系统属性”对话框中，完成如下图从 1 到 5 的一系列操作。新增的变量命名为 wxwin，值为 X:\wxWidgets-3.0.0。



接下来的设置要在 Code::Blocks 中进行。

打开 Code::Blocks，选择菜单 Settings->Global Variables...，在设置 default 下新建一个 wx 变量，在 Build-in fields: 下，base 中填入“\${wxwin}”（wxwin 是刚才设置好的一个变量），include 中填入“\${wxwin}\include”，lib 中填入“\${wxwin}\lib”，这些都是开发中需要用到的“环境”中的一部分。





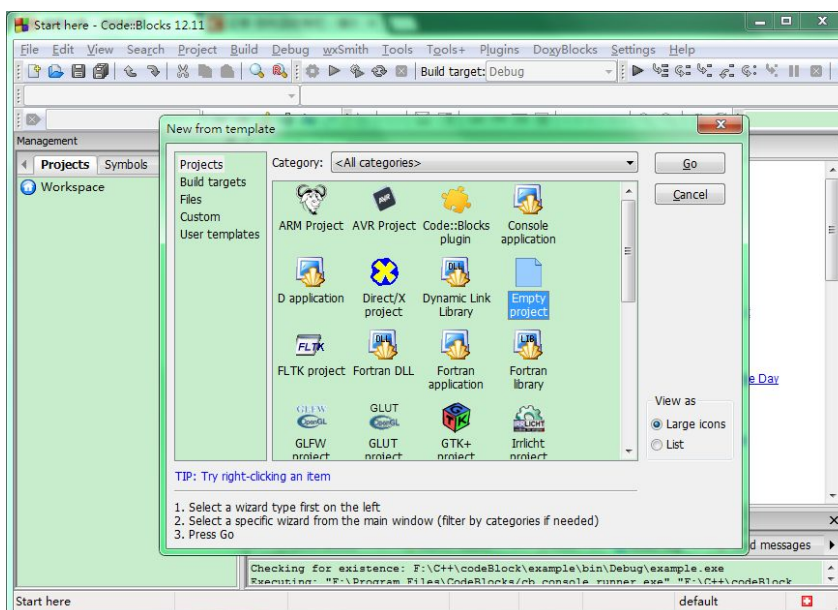
## 3.1 由“空项目”建立和运行 GUI 应用程序

下面将“白手起家”，由建立“空项目”开始，做一个简单的应用。程序改编自在线教程《wxWidgets tutorial》(<http://zetcode.com/gui/wxwidgets/>)的“First programs in wxWidgets”部分的第一个程序。这个教程，将作为建议学习方案中的主教程之一。

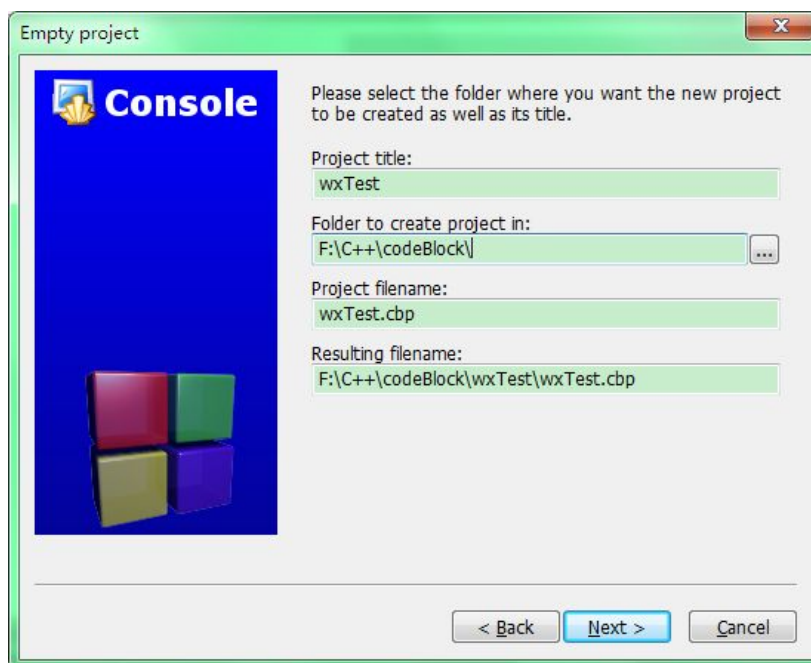
### 3.1.1 建立项目

建立项目的过程是：

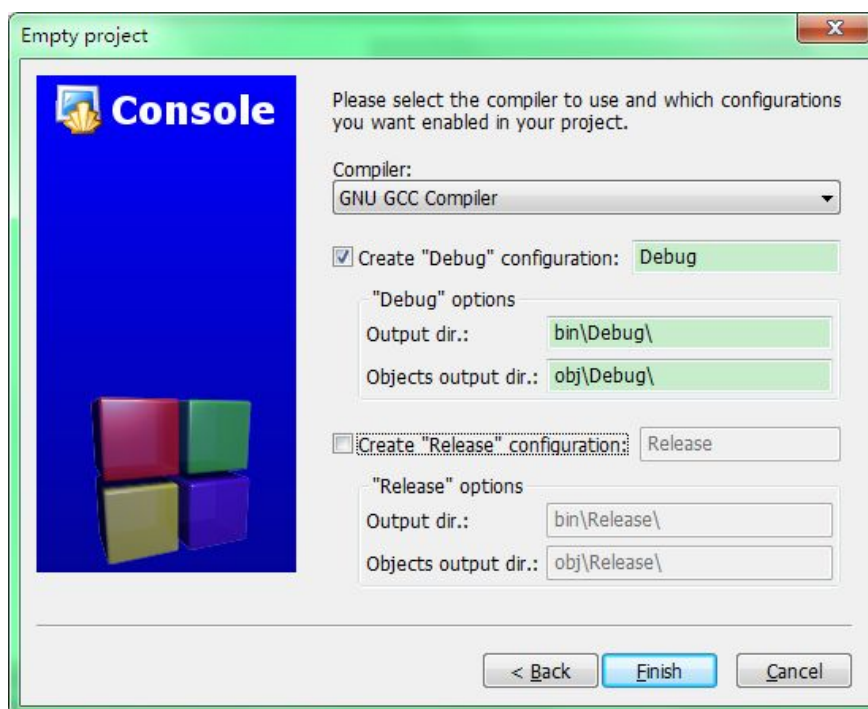
- (1) 通过菜单“File”->“New”->“Project...”，选择“Empty project”建一个空项目



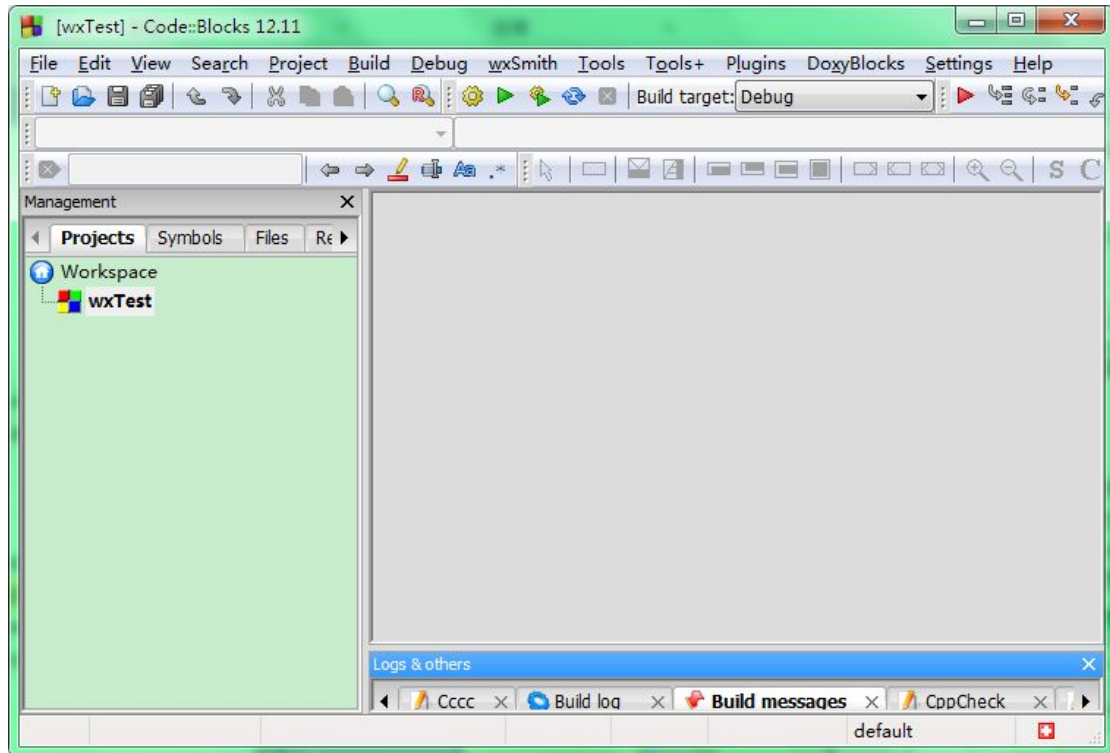
(2) 点击“go”按钮后，有一个欢迎界面，点击“next”，出现下图，填入项目名。我建立的项目名称为 wxTest。



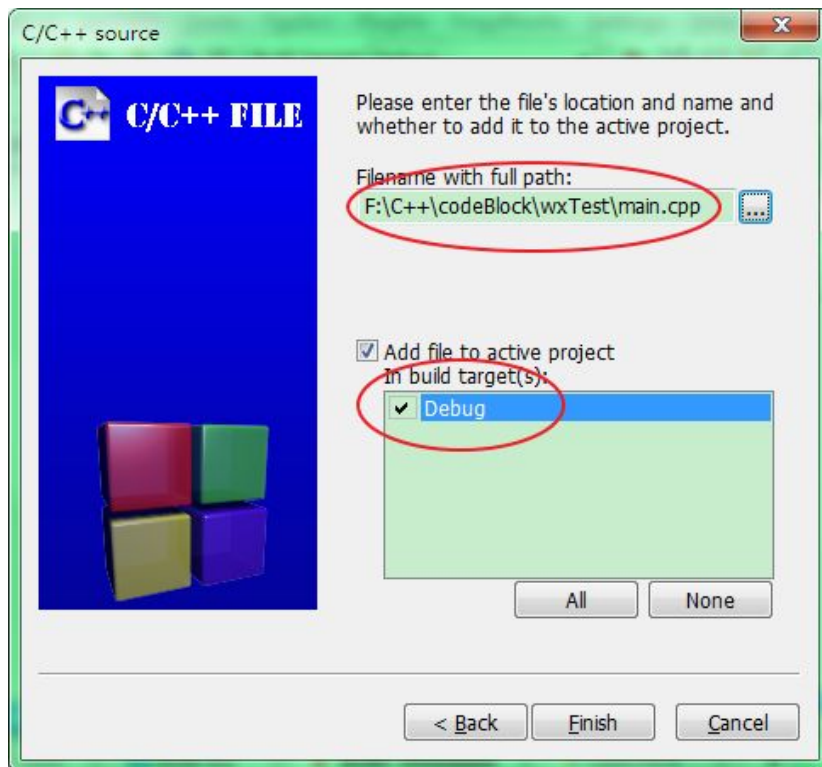
(3) 点击“next”后，要求选择编译器及生成的目标文件类型，照下图选择



(4) 点击“Finish”后，将生成一个空项目，如图



(5) 点击菜单 File->New->File...为项目新建一个源程序文件。在连续出现的几个对话框中，选择要增加的文件类型是“C/C++ source”（即源文件），再一个对话框中选择语言是“C++”。接着，在下图所示的对话框中，给出带完整路径的源文件名（本例中用 main.cpp），注意将 Debug 复选框选中。





(6) 点击“Finish”后，将下面的源程序输入（或粘贴）到文件 main.cpp 中。

```
#include <wx/wx.h>
class Simple : public wxFrame
{
public:
    Simple(const wxString& title);
};

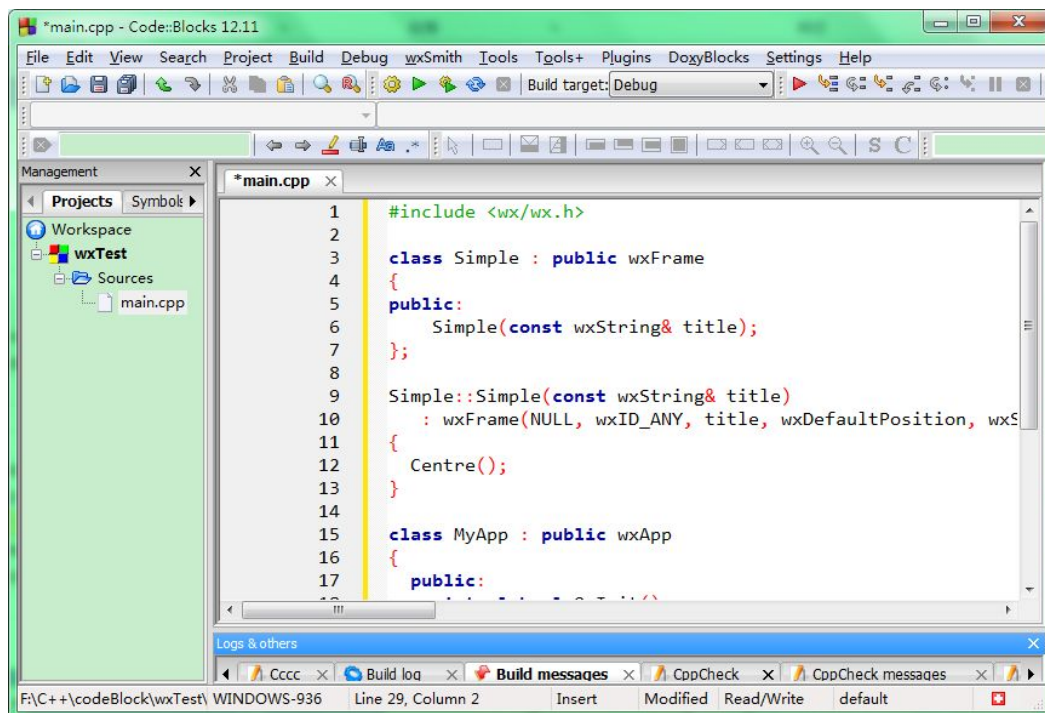
Simple::Simple(const wxString& title)
    : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 150))
{
    Centre();
}

class MyApp : public wxApp
{
public:
    virtual bool OnInit();
};

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{
    Simple *simple = new Simple(wxT("Simple"));
    simple->Show(true);
    return true;
}
```

加入了源代码之后的项目如下图所示：



可以暂时不考虑程序中的语句是什么意思。能完成运行程序的完整过程，是我们当前的任务。能运行程序了，后面再看“门道”。

下面将对这个项目进行编译，进而看到运行结果。

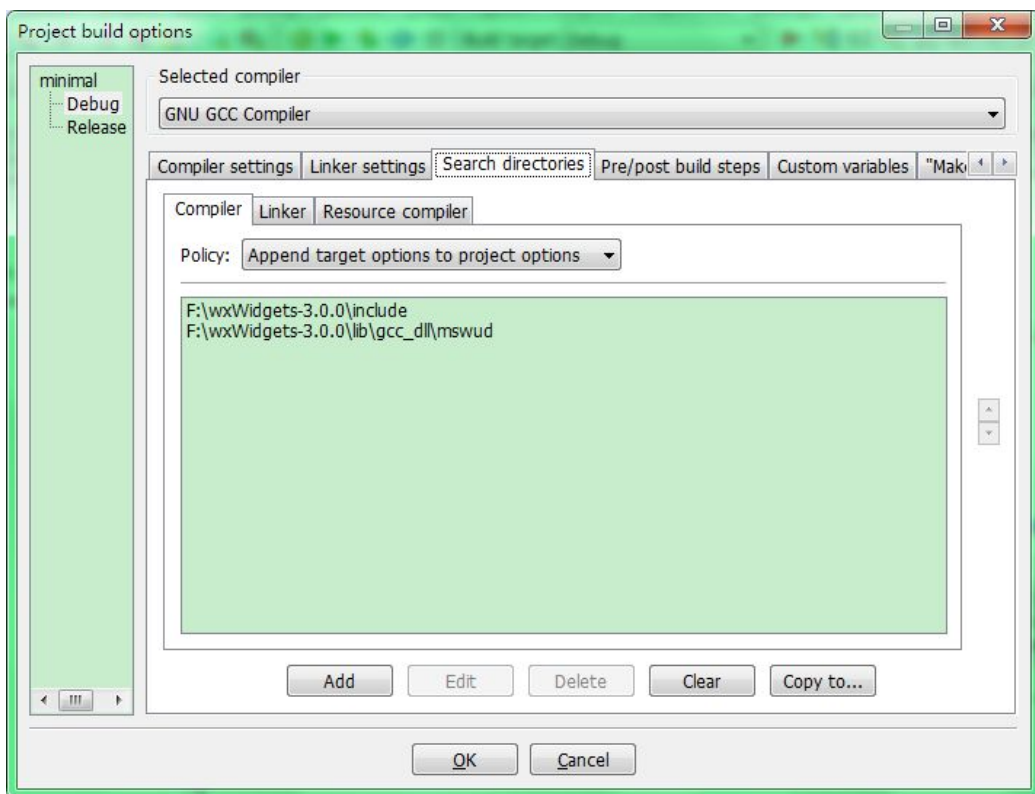
### 3.1.2 编译和运行项目

选择菜单“Build”中的“Build”选项（或者工具栏中的相应按钮）对项目进行编译、连接。程序第一行即出现错误。错误提示是：

```
fatal error: wx/wx.h: No such file or directory
```

也就是说，找不到要包含的头文件 wx/wx.h。

这需要设置“搜索路径”解决。选菜单 Project->Build options..., 在选项卡 Search directories 中，设置 Compiler。通过“Add”增加目录 X:\wxWidgets-3.0.0\lib\gcc\_dll\mswud 和 X:\wxWidgets-3.0.0\include，结果如下图所示：

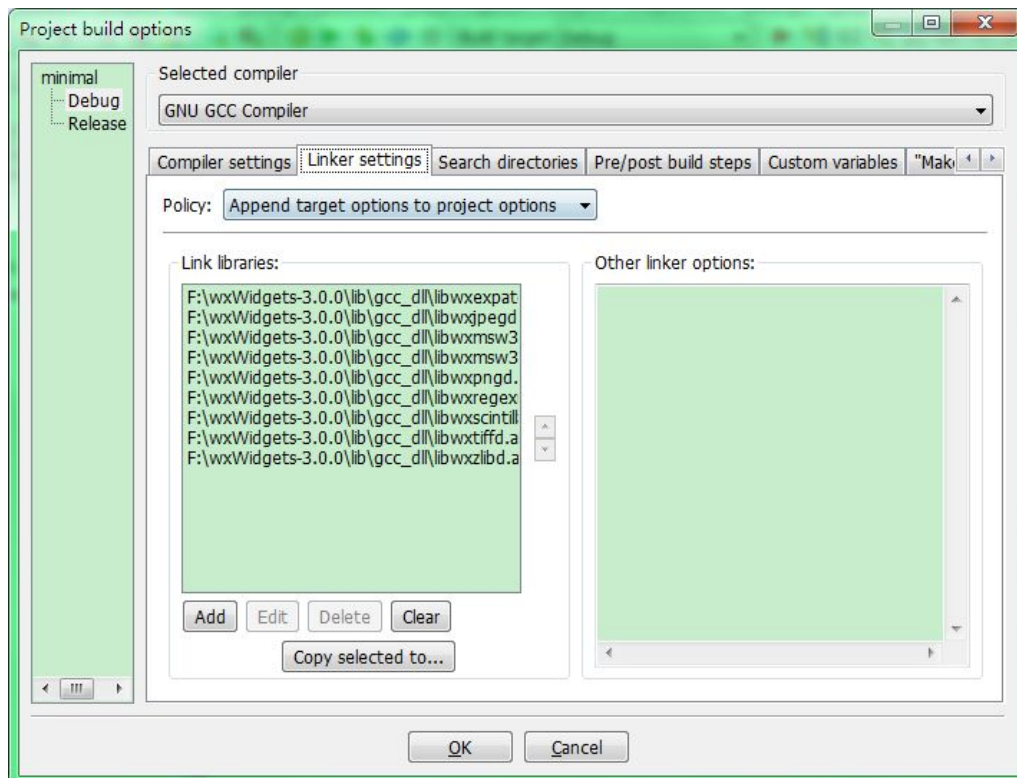


接着 Build，将不再有语法错误。

出现的一大堆错误提示，来自于连接环节，问题是找不到库文件。

选菜单 Project->Build options..., 在选项卡 Linker settings 中，需要加入要连接的“目标文件”。如图所示，通过 Add 按钮加入 X:\wxWidgets-3.0.0\lib\gcc\_dll 文件夹中的所有.a 文件

(实际上, 选择其中几个需要的就可以了。因为不知道究竟需要哪几个, 全选是最省事的办法):

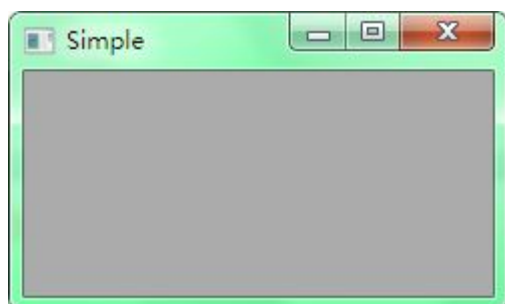


然后再编译, 0 errors, 0 warnings。成功!

但是运行程序时, 会出现错误, 如下图:



按提示来, 在 X:\wxWidgets-3.0.0\lib\gcc\_dll 中找到 wxmsw30ud\_gcc\_custom.dll 文件, 将其拷贝到项目所在文件夹, 再运行, 就看到了期盼的窗口, 如图所示。



出现上面的运行错误，原因是我在 Linker settings 加入的是 lib\gcc\_dll 文件夹中的.a 文件，这些属于“动态链接库”（这个术语自己百度去吧）。这种方式的好处在于编译速度快，目标代码小，但是在编译好的程序运行时，必须要能找到需要的.dll 文件。最简单的办法，就是拷贝.dll 文件。

这个程序很短，结果也只是一个空空的窗口，但是作为掌握编译、运行窗口程序的案例，却也是足够的了。

以上的设置和文件复制，“有经验”之后可以提前完成。上面的描述方法，是考虑到希望读者对各个环节的问题，能多些感觉。

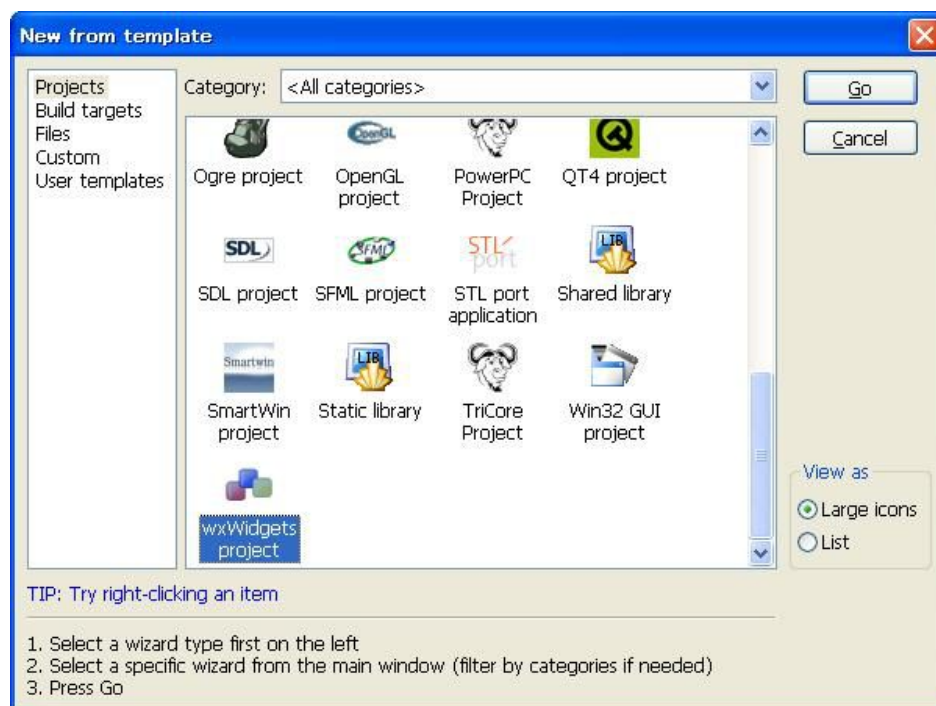
## 3.2 利用 Code::Blocks 的向导建立应用

另一种在 Code::Blocks 中建立 wxWidgets Project 的方法，是通过“向导”开发应用。这种方法用得不是很多，可以作为了解。

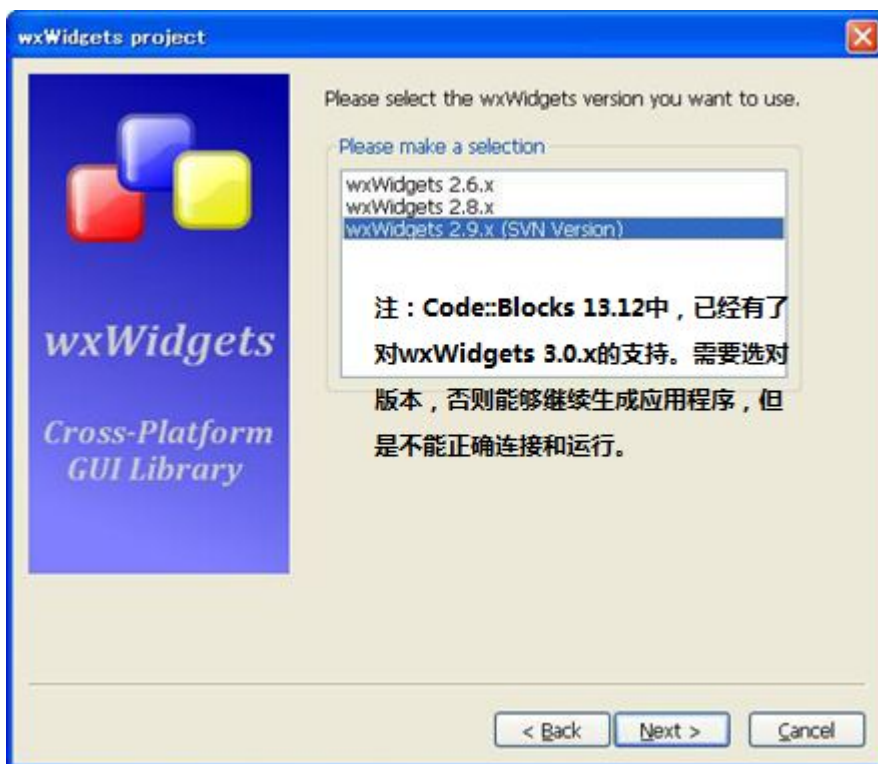
本节下面的材料，非作者原创，整理自 <http://www.cnzui.com/archives/962>。

利用向导开发的具体步骤是：

（1）通过菜单“File”->“New”->“Project...”，选择最后面的 wxWidgets project。

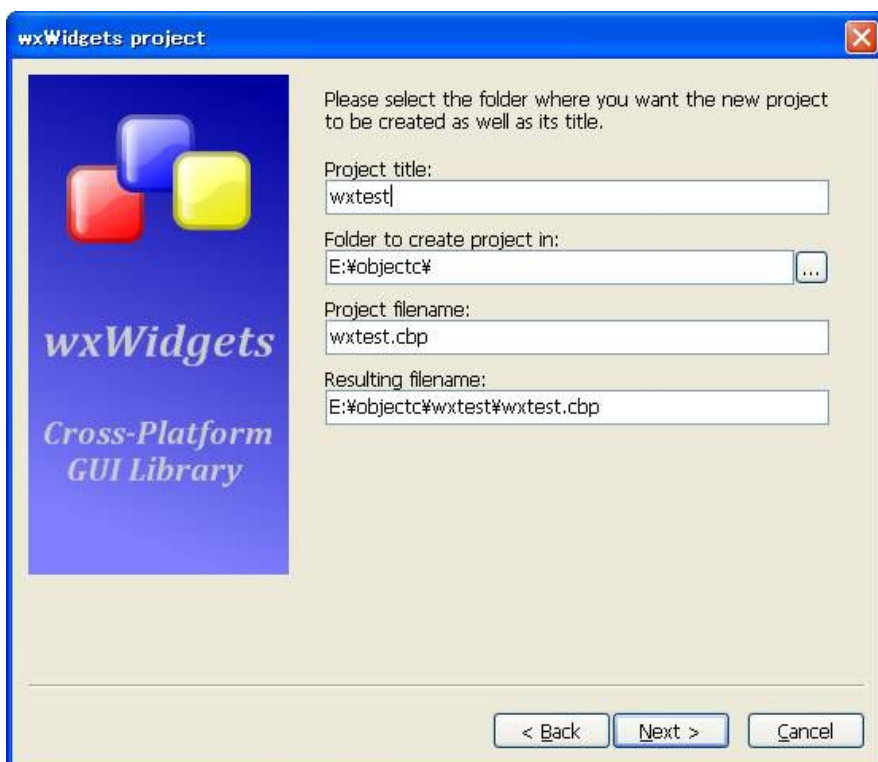


（2）点击“Go”进入工程配置向导，首先会出来一个欢迎窗口，直接 next 后，选择安装好 wxWidgets 版本。

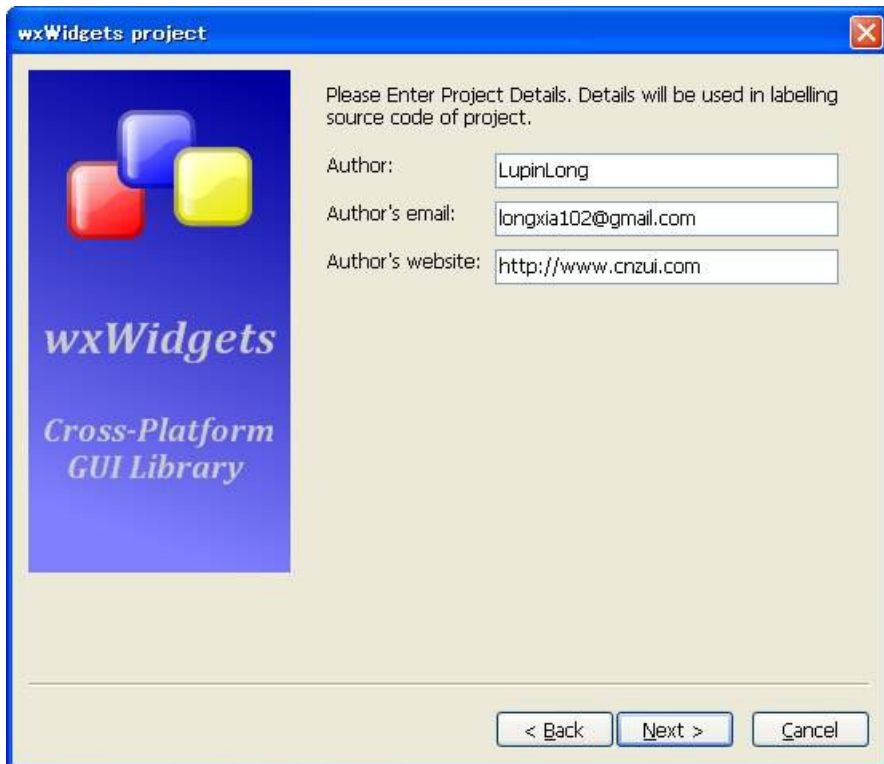


注：Code::Blocks 13.12 中，已经有了对 wxWidgets 3.0.x 的支持。需要选对版本，否则能够继续生成应用程序，但是不能正确连接和运行。

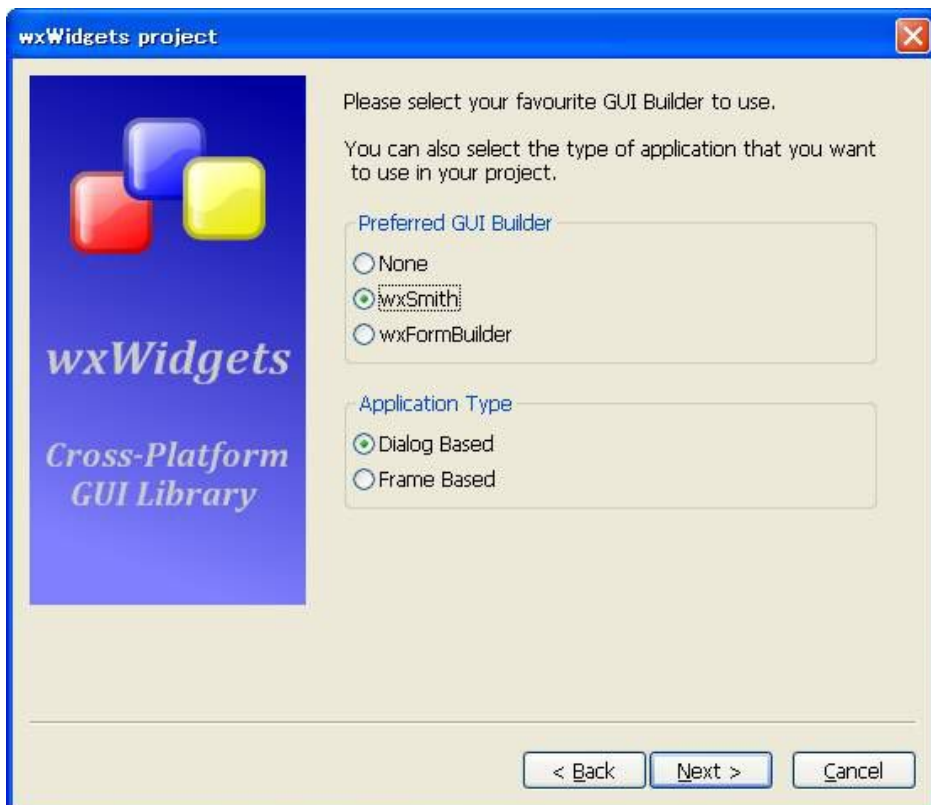
(3) 按“Next”，然后输入项目名“wxtest”，选择保存项目的文件夹。



(4) 继续“Next”，输入作者和及一些版权说明信息。

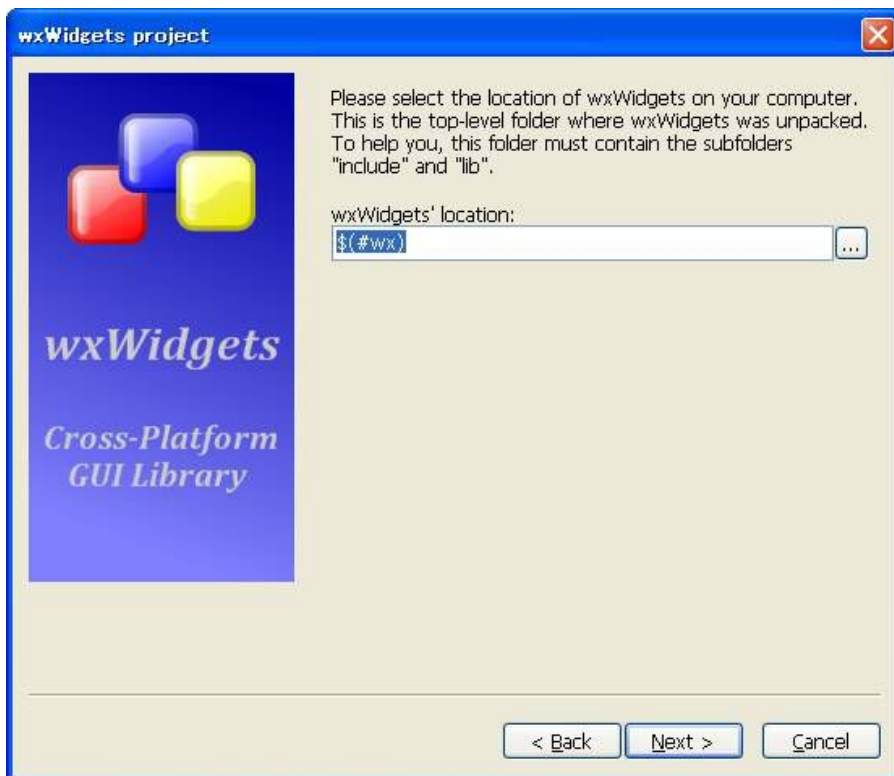


(5) 继续“Next”，选择 GUI 设计工具和程序类型，用 wxSmith 和 Dialog based。

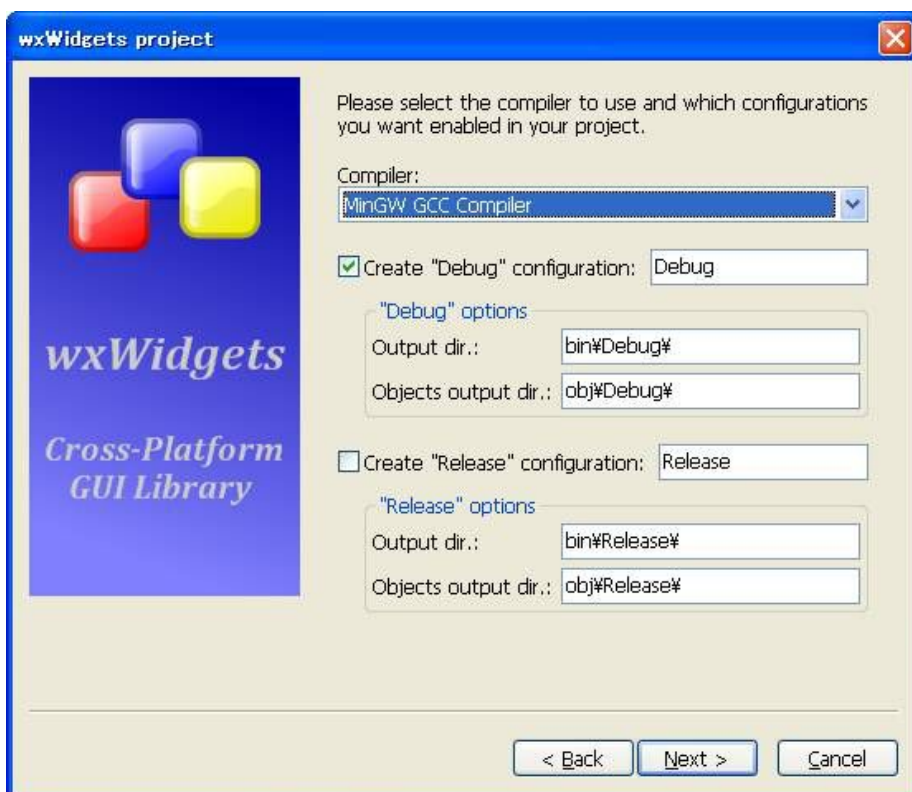


(6) 按“Next”，接下来的就是 wxWidgets 环境的一些设置了，这里我们输入刚才设置的 wxWidgets 根目录，直接填入“\$(#wx)”就可以了。

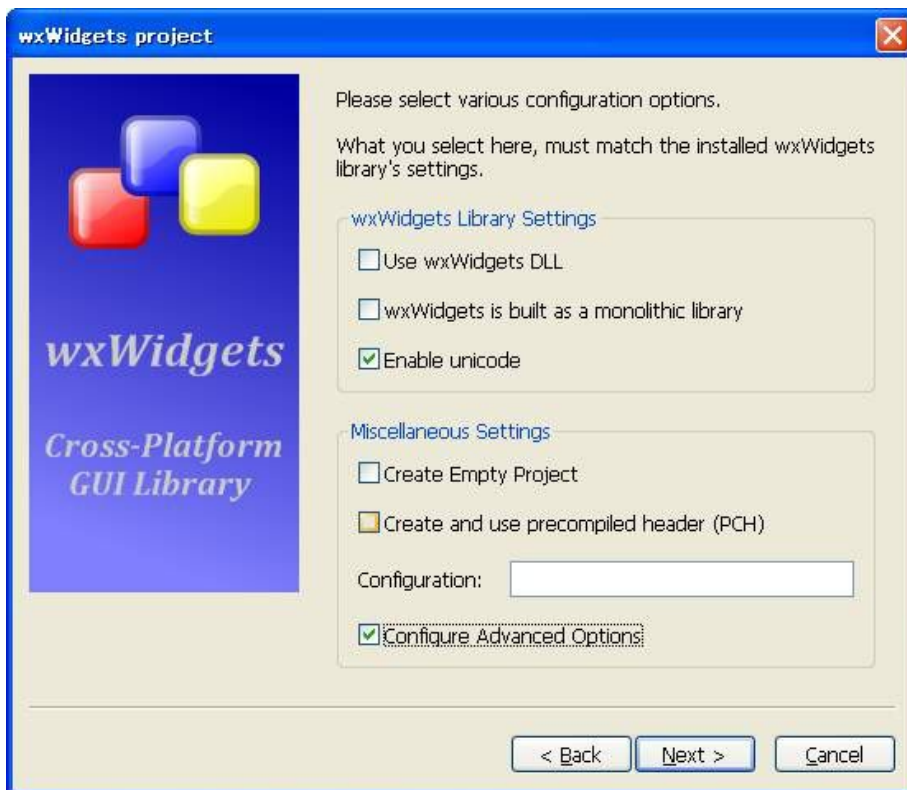




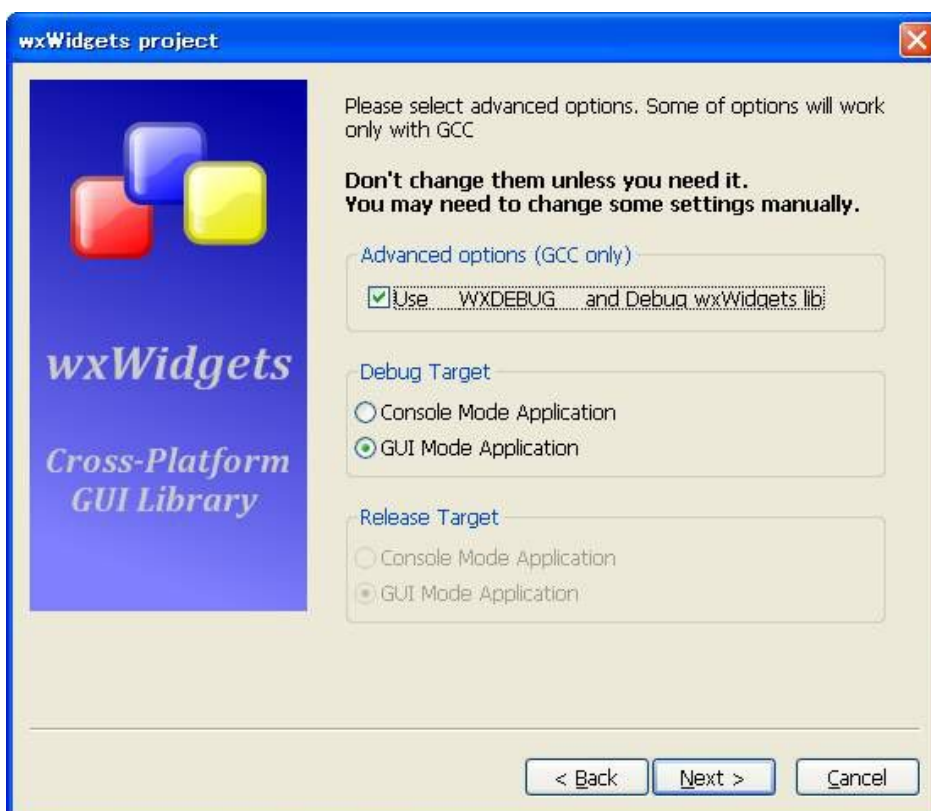
(7)按“Next”，这里我们看到默认就是选择了 MinGW 编译器了，下面的我们只做 Debug 版本，所以只选上“Create “Debug” configuration”。



(9)继续“Next”，接下来要选择怎么使用 wxWidgets 库，这里根据你编译的 wxWidgets 库是什么样的来。

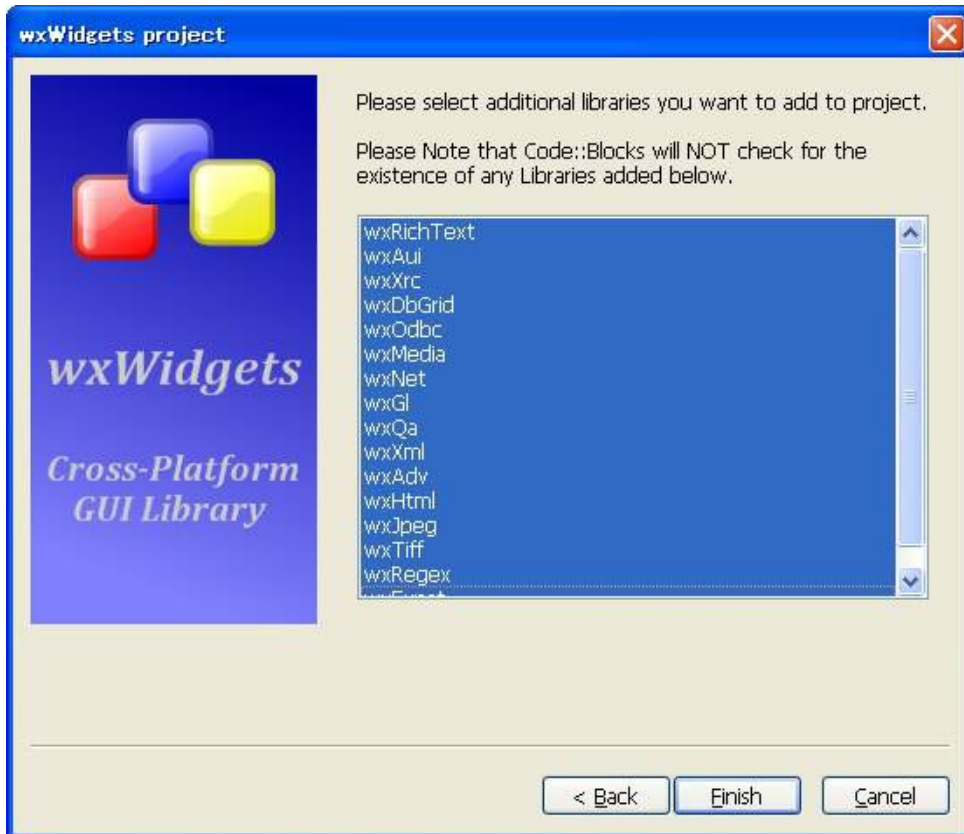


(10) 按“Next”，因为选择了“Configure Advanced Options”，所以要对使用 wxWidgets 库做更多的设置，这里我们选上我们使用 lib 方式调用（要求编译 wxWidgets 时用 SHARED=0 参数，生成了静态库文件）。



(11) 按“Next”进到最后一步，我们选择需要用到的库，不知道的话全部选上。





(12) 点击“Finish”，工程随即建立成功。

这时可以查看项目中自动生成的文件，其中有.cpp 的源文件，也有.h 的头文件。再细读，和 3.1 中输入的程序长得差不多。

其实，向导的作用，就是通过一系列的选择，由向导程序自动生成应用程序。

如上步骤创建的应用程序的运行的结果是：



在编译和运行时，都有可能出现一些错误。这一般不是程序本身的问题，而是 Code::Blocks 的编译环境和运行的支持文件不全而造成的。

请参阅 3.1.2 小节，可能会帮助你排除问题，让程序正确运行。

## 4 wxWidgets 学习资料及利用方法指导

初学者常苦于找不到参考资料。实际上，是找不到，不是没有。真正有用的资料，常常也就在手边，只是不知道。有能力熟练地使用一切能用得着的资料，这是水平提高的指标之一。这种能力，同样，也是在实践中获得，而不是有谁为你讲一堂课就能得到。

本章的学习资料，从最一般的——书籍开始谈起。

### 4.1 关于 C++ wxWidgets 的书籍

#### 4.1.1 《使用 wxWidgets 进行跨平台程序开发》

关于 wxWidgets 的书籍还真少见，从亚马逊上只查到一本《使用 wxWidgets 进行跨平台程序开发》(<http://www.amazon.cn/gp/product/B00A1WDQ30>)，部分电子版（含书中例程的源代码，从 <http://download.csdn.net/detail/cjylg/2997827> 下载）。看得好，请支持纸质出版。

这本书的英文版叫《Cross Platform GUI Programming With wxWidget》（见 <http://www.wxwidgets.org/docs/book/>），我浏览过其中的一部分，读起来不难。

#### 4.1.2 wxwidgets 的 Wiki 主页

我推荐阅读的是 wxwidgets 的 Wiki 主页 ([http://wiki.wxwidgets.org/Main\\_Page](http://wiki.wxwidgets.org/Main_Page)) 中的 Guides & Tutorials 部分 ([http://wiki.wxwidgets.org/Guides\\_%26\\_Tutorials](http://wiki.wxwidgets.org/Guides_%26_Tutorials)) 链接的一个教程《wxWidgets tutorial》(<http://zetcode.com/gui/wxwidgets/>)。写作时再次看 wxwidgets 的 Wiki 主页，发现其中的宝，太多了。开源社区的贡献者不仅提供软件的共享，而且将这种共享精神延续到指导书籍，必须赞。

#### 4.1.3 《wxWidgets tutorial》

《wxWidgets tutorial》是我极力推荐的一个在线教程。其中的叙述很少，一直在用小例子，启发读者获得对 wxWidgets 的认知。我一边读代码，一边练习，完成了一次愉快的学习

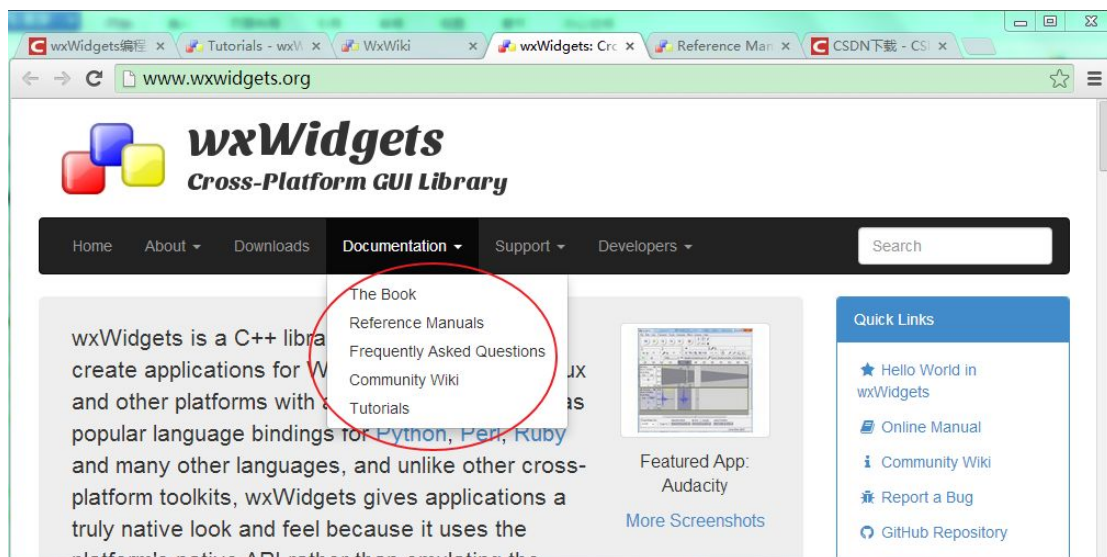
之旅。《wxWidgets tutorial》最后一章开发了一个俄罗斯方块游戏，当看完并同步练习完之时，也就是掌握了用 wxWidgets 开发应用程序的基本路数之时。

为方便读者，我将这个教程中整理到一个 Word 文档中，作为本教程的一个附件，一起打包供下载使用。在 Word 文档中，加入了部分批注，是我在学习中文档中查阅得到的线索。

唯一引发本文读者不爽的是，《wxWidgets tutorial》是英文版的。这其实是个好事，在这个时代，靠着翻译来的二手中文书还想学到新技术，不拍脑袋也知道这只是天方夜谭的事。不断学习英语，是 IT 学生的学习形态。我一直鼓励同学们“在用英语中学英语”（见 <http://blog.csdn.net/sxhelijian/article/details/12177147>），这就是一个大大的时机。是否能看下去，不决定于你英语水平的高低，而是决定于你的心态。再进一步，这本书中的描述性文字很少，即使高考英语时是在考场抓阄决定 ABCD 的，也能看下去，只要去看。

## 4.2 用好 wxWidgets 的在线文档

在互联网时代，另一类资料必须引起学习者的注意，那就是在线的文档和教程。到 wxWidgets 的主页 <http://www.wxwidgets.org/> 中看看，其中 Documentation 部分的每一个链接，各自都连接着一座宝库，如下图：



### 4.2.1 成熟平台常有在线文档

用微软平台开发程序时，最好的参考是 MSDN (<http://msdn.microsoft.com/library/>)，而用 Java 开发时，有 Java SE 6 Documentation (<http://docs.oracle.com/javase/6/docs/index.html>)，

也有部分内容被翻译成中文 (<http://www.javaweb.cc/JavaAPI1.6/>)。在软件开发过程中,需要的类、函数、宏是记不住的。有在线文档查找,专业人员不记这些。大多数成熟的平台,既提供真正在线的文档,这些文档也可以下载到本地,通过浏览器阅读。

## 4.2.2 wxWidgets 的在线文档

wxWidgets 也有在线文档 (<http://docs.wxwidgets.org/3.0/>), 在 2.1 节, 要求读者下载了 Manual(HTML).zip。

现在请选择一个自己用着习惯的文件夹, 将 Manual(HTML).zip 解压缩。我解压缩到了 F:\wxWidgets-3.0.0-docs-html。找到其中的 index.html 文件, 双击打开, 如下图所示, 这就是将有大用的最佳参考。



建议用鼠标右击 index.html 文件, 在菜单中选择“发送到->桌面快捷方式”。在桌面上建立打开在线文档的快捷方式, 将方便以后的使用。

### 4.2.3 查找在线文档

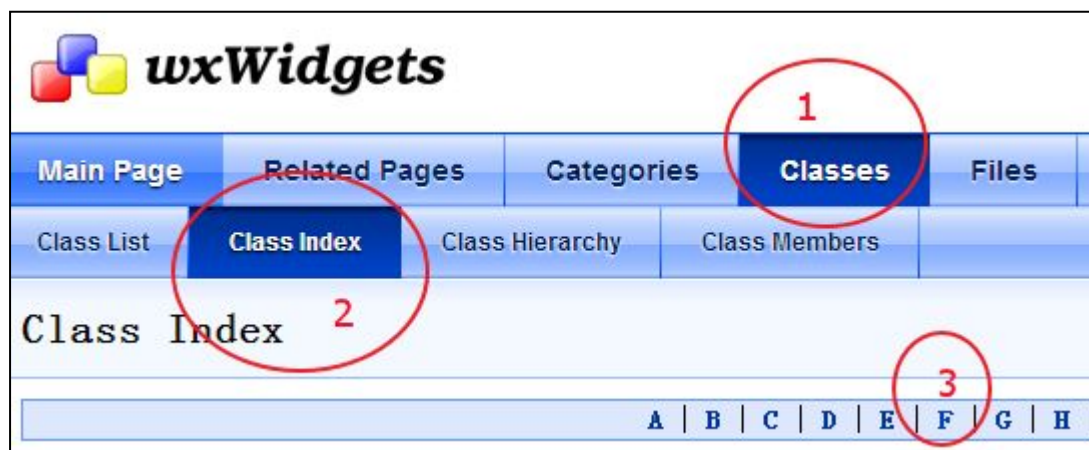
在线文档通过超链接组织起相关材料之间的联系。作为实践，将各个链接点一点，你会有感觉。

举一个例子。在《wxWidgets tutorial》中，Menus and Toolbars 部分第一节 Simple menu example 中的例程，有一段如下代码：

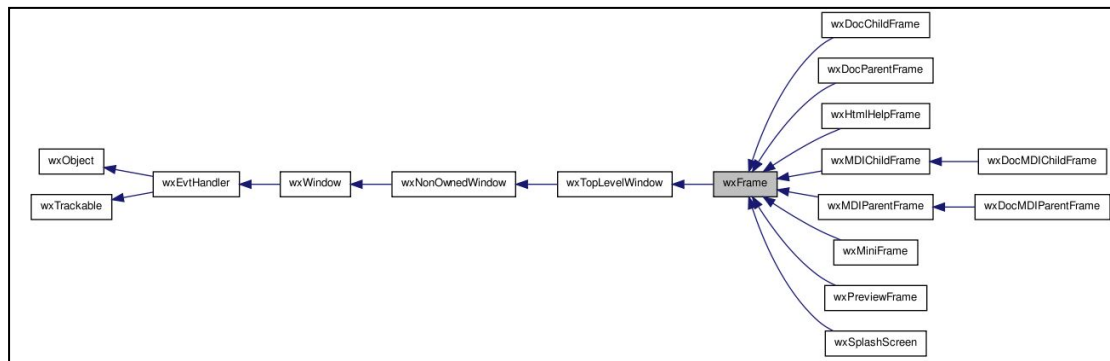
```
SimpleMenu::SimpleMenu(const wxString& title)
    : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(280, 180))
{
    menubar = new wxMenuBar;
    file = new wxMenu;
    file->Append(wxID_EXIT, wxT("&Quit")); //练习：想知道 Append 函数的情况
    menubar->Append(file, wxT("&File"));
    SetMenuBar(menubar);

    Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
            wxCommandEventHandler(SimpleMenu::OnQuit));
    Centre();
}
```

现在想知道 `wxFrame` 类的构造函数中的各参数含义，要点的链接是：Class->Class Index ->F（wxWidgets 中类的命名均以 wx 开头，取 `wxFrame` 中的 F），如下图：



接着，在“wxFrame Class Reference”页面，可以看到 `wxFrame` 类同其他类的继承关系以及其他信息，如下图：



我们关心 `wxFrame` 类的构造函数，继续往下看，可以在“Constructor & Destructor Documentation”部分看到构造函数的定义及说明：

```

wxFrame::wxFrame (    wxWindow *    parent,
                    wxWindowID    id,
                    const wxString &    title,
                    const wxPoint &    pos = wxDefaultPosition,
                    const wxSize &    size = wxDefaultSize,
                    long    style = wxDEFAULT_FRAME_STYLE,
                    const wxString &    name = wxFrameNameStr
                    )
  
```

Constructor, creating the window.

### Parameters

<b>parent</b>	The window parent. This may be <b>NULL</b> . If it is non- <b>NULL</b> , the frame will always be displayed on top of the parent window on Windows.
<b>id</b>	The window identifier. It may take a value of -1 to indicate a default value.
<b>title</b>	The caption to be displayed on the frame's title bar.
<b>pos</b>	The window position. The value <code>wxDefaultPosition</code> indicates a default position, chosen by either the windowing system or <code>wxWidgets</code> , depending on platform.
<b>size</b>	The window size. The value <code>wxDefaultSize</code> indicates a default size, chosen by either the windowing system or <code>wxWidgets</code> , depending on platform.
<b>style</b>	The window style. See <code>wxFrame</code> class description.
<b>name</b>	The name of the window. This parameter is used to associate a name with the item, allowing the application user to set Motif resource values for individual windows.

这里，构造函数的参数、返回值一目了然，各个参数的含义、用法、默认值也可以看到。这是最佳的第一手的开发参考资料。

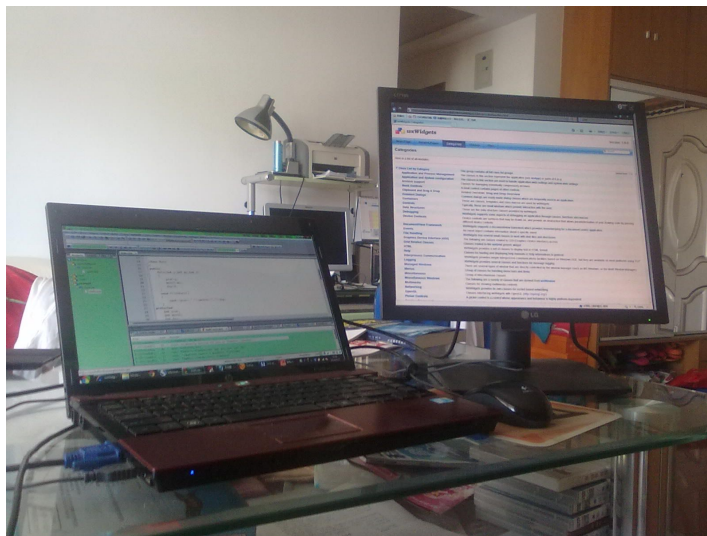


下面请做一个练习。在读下面的程序时，想知道 `Append` 函数的情况。如何利用在线文档找到说明？请找到后，再找一遍真正学会。

关于上面的代码中还有哪些疑问，试着通过在线文档给出解答。

## 4.2.4 查看在线文档的设备支持

如果有条件，布置如下图所示的工作条件。用双屏，一个看文档，一个写程序。



当前的学习，有很多时候用电子版的参考资料。开发中查阅在线文档，是件相当频繁的事情，双屏幕的配置成为需要。

当然，作为在校学生，当场地受限时，也不必太纠结这样的条件了。毕竟，这不是必须。

## 4.3 在编程环境中找帮助

本文使用的编程环境是 `Code::Blocks`。C++编码规范中，将类声明、常变量声明、宏定义等保存在头文件（.h）中，而将类、函数的实现用源文件（.cpp）保存。头文件实际上就是一个非常好的帮助文档。符合规范要求的开发者，通过恰当的命名，总能够让程序的阅读者“见文知义”，从头文件中得到足够的信息。

在 `Code::Blocks` 中，为找到这些信息提供了足够的支持。其实，其他 IDE，也能做到这一点。

例如，对于 4.2.3 中的那一段代码：

```
SimpleMenu::SimpleMenu(const wxString& title)
: wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(280, 180))
```

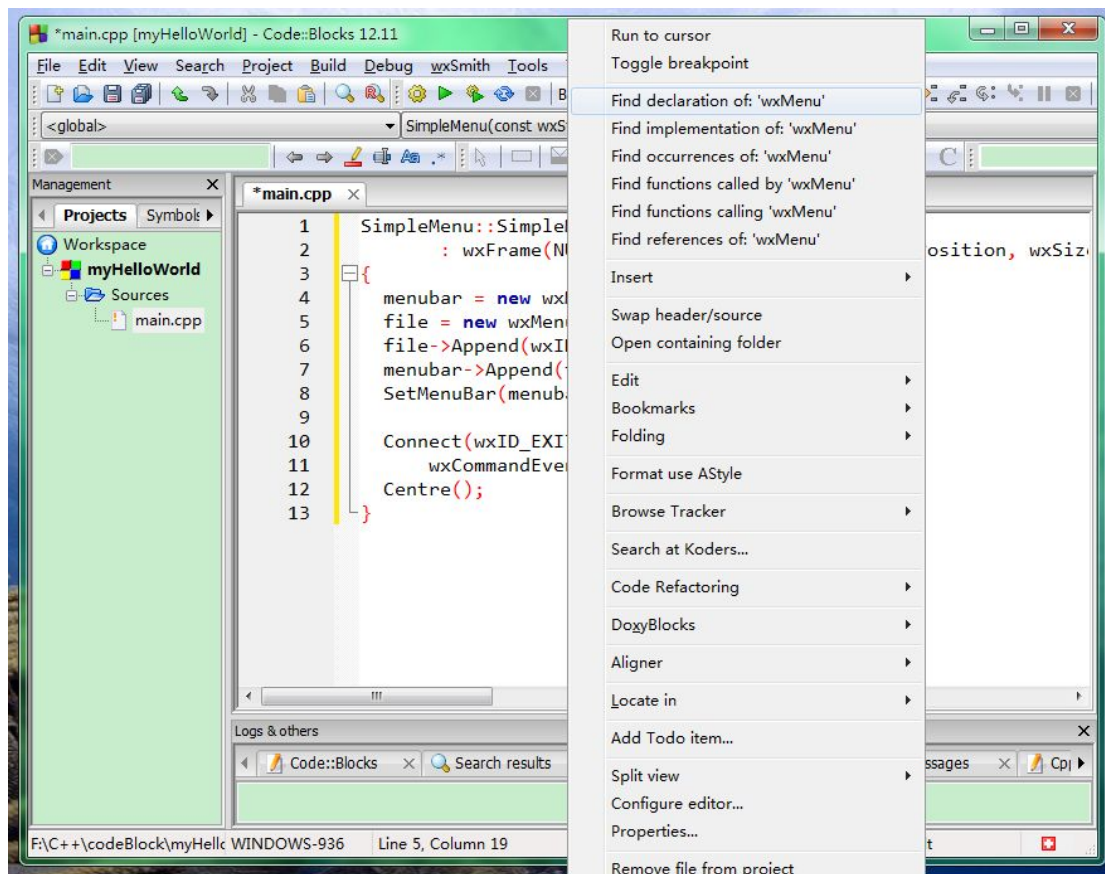
```

{
    menubar = new wxMenuBar;
    file = new wxMenu;
    file->Append(wxID_EXIT, wxT("&Quit")); //例
    menubar->Append(file, wxT("&File"));    //练习
    SetMenuBar(menubar);

    Connect(wxID_EXIT, wxEVT_COMMAND_MENU_SELECTED,
            wxCommandEventHandler(SimpleMenu::OnQuit));
    Centre();
}

```

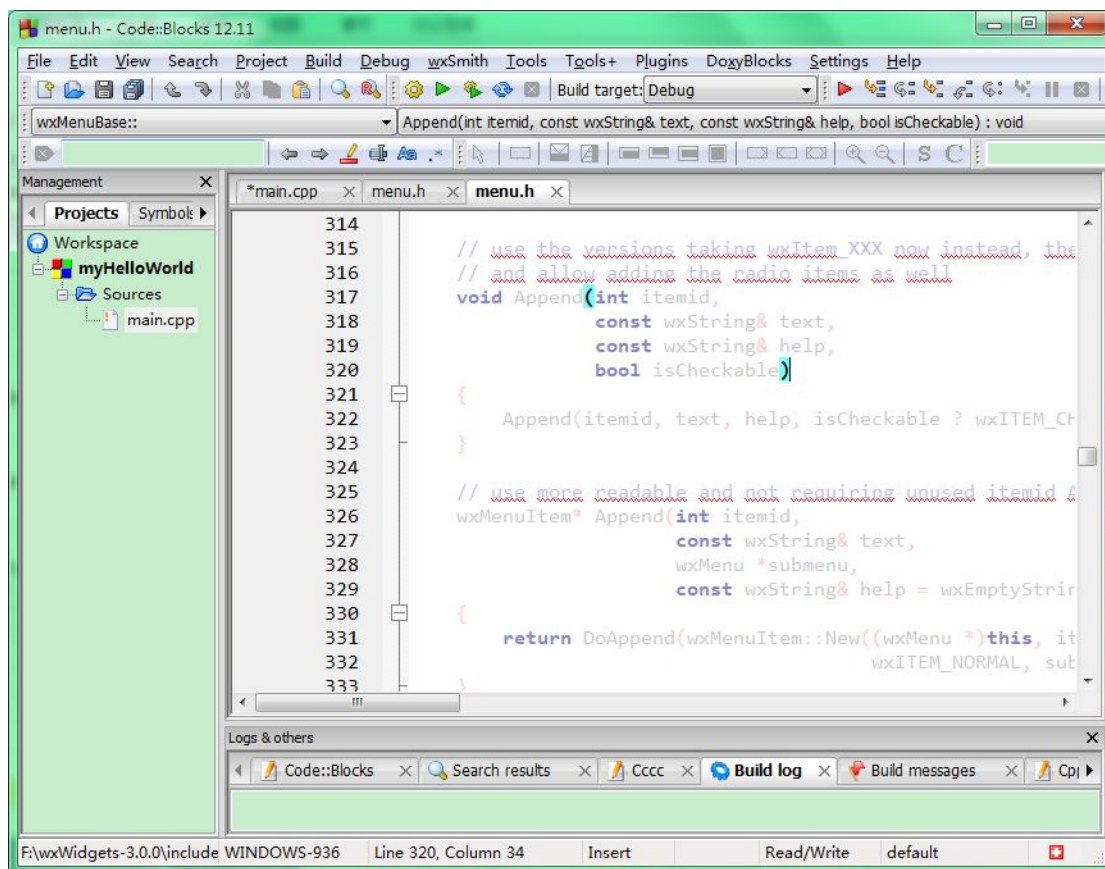
现在，想知道 `file->Append(wxID_EXIT, wxT("&Quit"));` 一句中 `Append` 函数各参数的含义。由上一句可以看出，`file` 是 `wxMenu` 类的实例，只要看 `wxMenu` 类的声明即可。于是，在 Code::Blocks 中，单击鼠标右键，单击在菜单中出现的“Find declaration of: `wxMenu`”选项，如图：



在 Code::Blocks 中，将打开新的页签，显示 `wxMenu` 类的声明。但是 `wxMenu` 类并没有 `Append` 这样的成员函数，发现 `wxMenu` 类是 `wxMenuBase` 类的派生类，推断 `Append` 应该是继承自 `wxMenuBase` 类的成员函数。用相同的操作，在 `wxMenuBase` 类名上单击鼠标右



键并选择对应的选项，在新出现的页签中，找到了 `Append` 成员函数的 4 种重载形式的声明，如下图所示。



现在做一个练习。找出 `menubar->Append(file, wxT("&File"))`; 中 `Append` 函数的声明，从中看出其调用的方法。

## 4.4 深入学习路线建议

### 4.4.1 看书的策略

在完成本文前 3 章的工作之后，建议同时看《使用 wxWidgets 进行跨平台程序开发》和《wxWidgets tutorial》这两本书，同步地实践书中的程序。

同步看，意味着交叉、重复再看。可以先运行例程，有感性认识之后再阅读代码。一次看不明白不要紧，继续往后看，或者看另外一本书的相关部分，当再次看时，问题或许能够自然化解。

我在学习中，看完了《使用 wxWidgets 进行跨平台程序开发》的前两章，然后将《wxWidgets tutorial》从头看到完，再看《使用 wxWidgets 进行跨平台程序开发》中的后面部分时，常能联系起《wxWidgets tutorial》中运行过的例子。这种安排的体验，感觉不错。

在 Code::Block 中频繁新建项目很麻烦，况且每建一个项目还得设置 Build options...。我的做法是，建立了一个只包含一个源文件的项目。所有的练习，都是将代码粘贴到这个文件中完成，这节约了不少时间。

《wxWidgets tutorial》中的例子写得非常规范，严格执行了.h 头文件中写声明，.cpp 源文件中写实现的要求。我在实践时，偷了个懒，将本应放在多个文件中的内容，粘贴到前述的一个文件中。当然，诸如

```
#include <wx/wx.h>
#include <wx/menu.h>
```

之类的包含头文件要保留，而

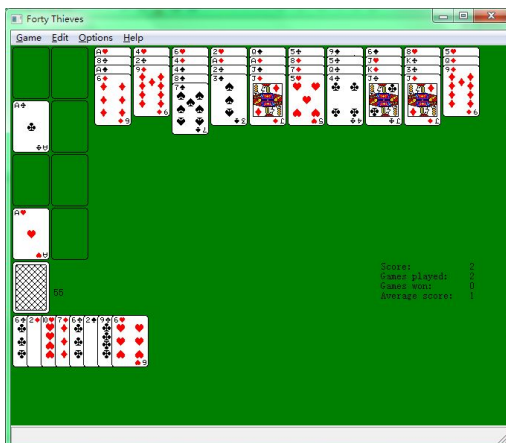
```
#include "menu.h"
```

之类的自定义头文件的包含命令，需要删除。因为合并到了同一个文件中，这些头文件根本不存在。

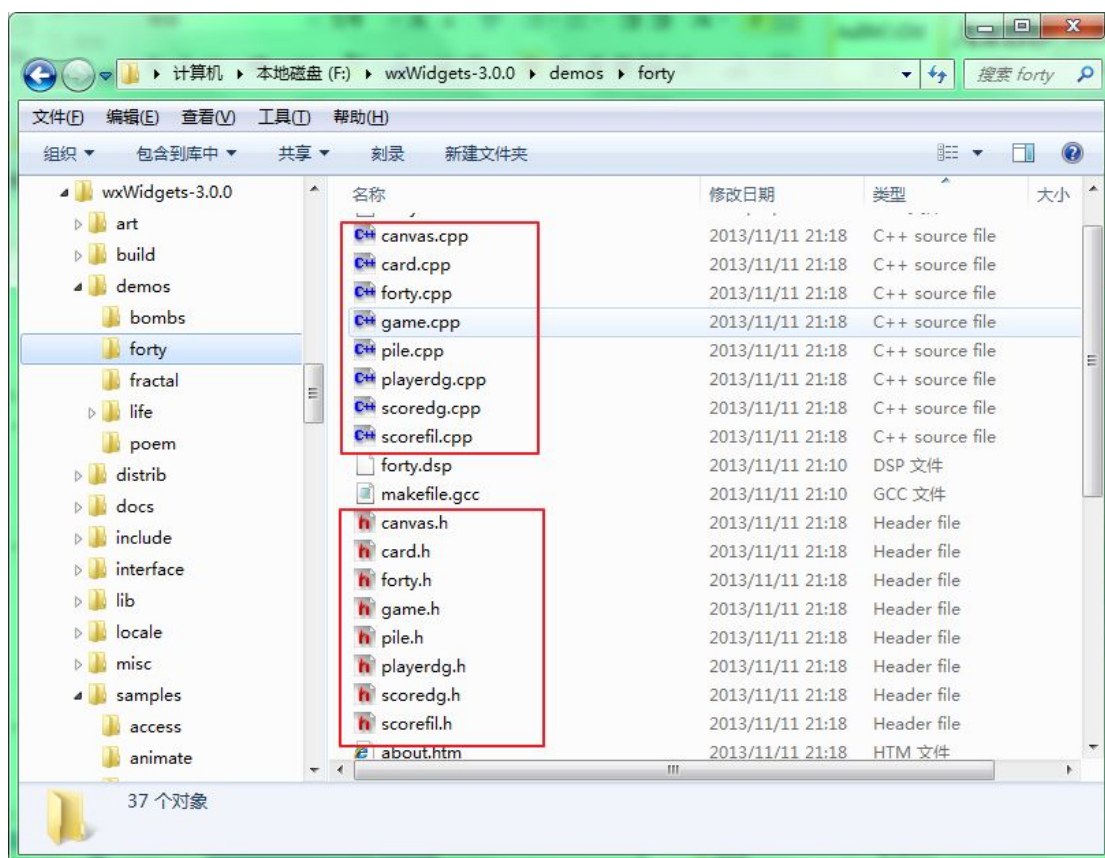
## 4.4.2 更多的案例

安装好的 wxWidgets 中还提供了很多演示和示例的项目。X:\wxWidgets-3.0.0\demos 中是所有的演示案例，示例的项目在 X:\wxWidgets-3.0.0\samples 中，学习到一定时候，运行这些程序，读一读源码，是一种很好的学习方式。

例如，wxWidgets 的 Demo 中的一个项目 forty，是一个纸牌游戏，运行结果如下图：



这个项目中的源文件如下图所示：



在 Code::Block 中新建一个项目，将 Demo\forty 中所有.cpp 和.h 文件复制到项目所在文件夹中，并通过鼠标右击项目名，在弹出菜单上选 Add file...的方式将文件加入项目，成为项目的源文件和头文件。Demo 为适应多种平台，提供了很多的文件，一般只需要.cpp 和.h 文件即可，如上图中加了方框的部分。

如果项目中有.xpm 文件（并不是每个项目都有），也请将这种文件复制过去，这是一种图形格式文件，程序中一般会用到。项目 forty 中就有 3 个这样的文件。

在更极少数情况下，还可能有其他文件需要复制过去。这可以通过读代码，看源程序中是否写了这个文件名。偷懒的办法，运行程序，若由于找不到文件出错了，会提示还需要哪个文件。比如项目 forty 中的 about.htm 文件。

后两类文件复制过去即可，是为支持程序运行的，不必通过 Add file...将其加入项目。

运行其他项目，方法类似。

## 5 用 wxSmith 进行可视化设计

按照第 4 章的要求看过了相关的书籍，应该具备了通过直接写代码的方式界面程序的能力。而在实际的工程开发中，做界面的工作常通过可视化的操作完成。wxSmith 就是这样一个支持 wxWidgets 快速开发的一个工具，专门用来做界面。

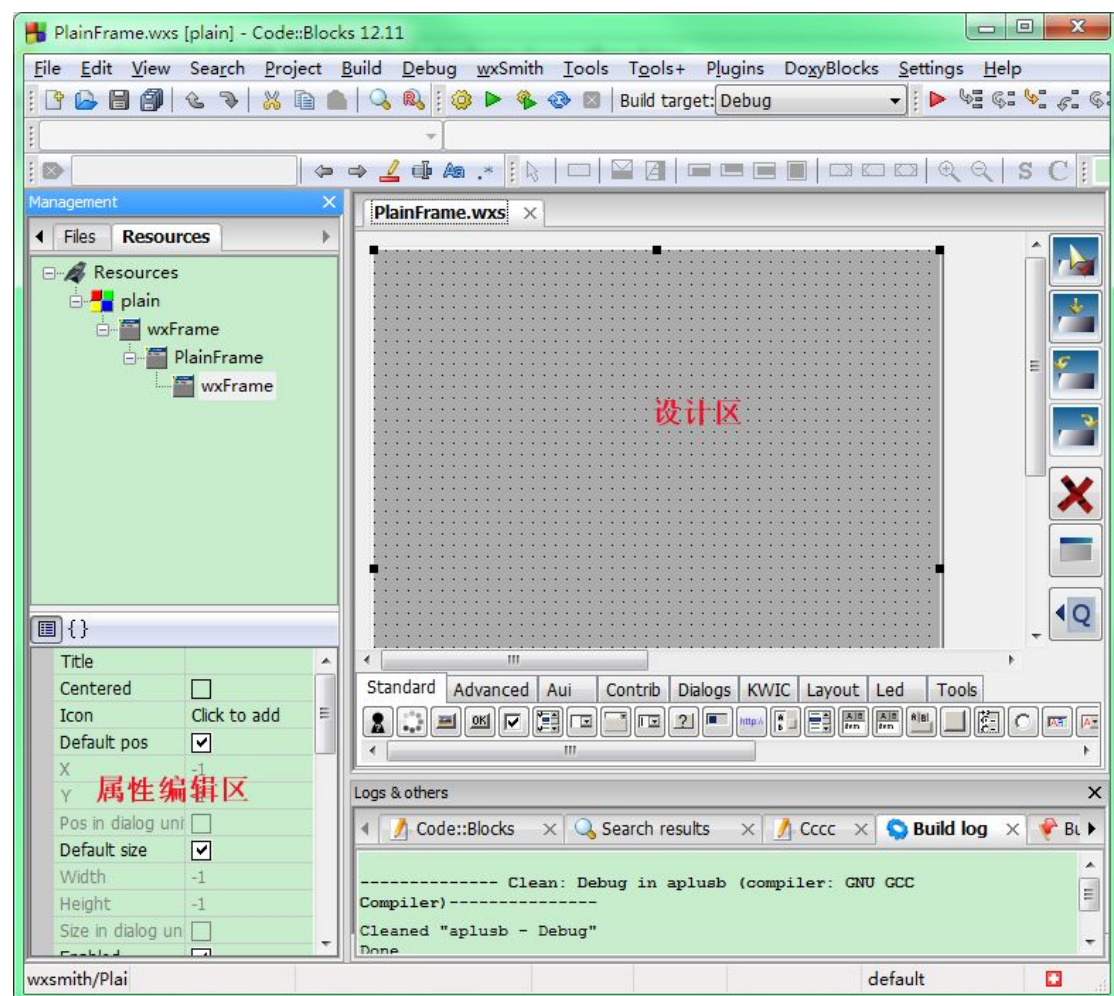
wxSmith 已经是 Code::Blocks 的一个标准配置，在 Code::Blocks 菜单中可以看到这一项。

这一章，我们将从做一个应用程序开始。应用程序的窗口中有两个按钮，按 SayHello 按钮，屏幕上会出现“Hello World”，按 Quit 按钮，程序将结束。



### 5.1 用 wxSmith 创建应用程序的外观

创建一个空项目，名称为 plain。选择菜单 wxSmith -> Add wxFrame，将要建立的 Frame 命名为 plainFrame。

在左侧的 Management 窗口中，显示 Resources 页签，我们看到的界面如下图所示：

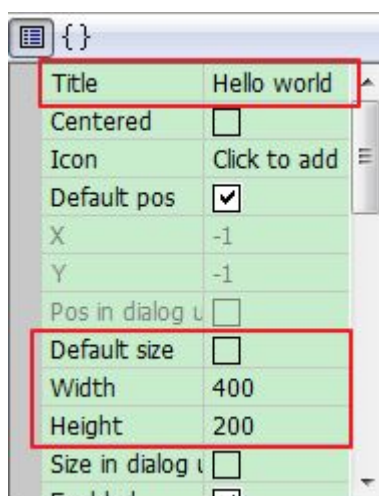


窗口中由许多小点构成的部分，就是将来应用程序的界面部分，我们称之为“设计区”。

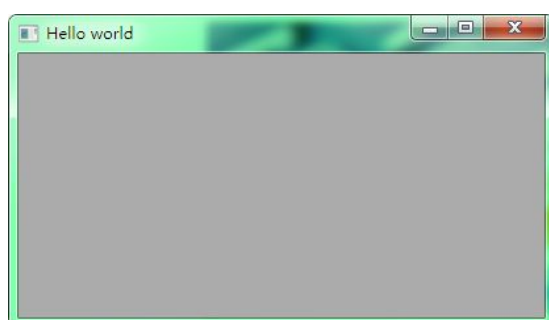
在整个窗口的左下方，是“属性/事件”窗口，被称为“属性编辑区”在其中可以设置属性，以及加入响应“OnClick”之类事件的代码。点击靠左的显示和设置“属性”，而右边的显示和设置“事件”（现在动手试一下）。

现在只看到整个 Frame 的属性和事件，我们做些修改。

例如，将 Default size 属性后面的“√”取消掉，将随后的 Width 和 Height 分别设置为 400 和 200。再将用来设置窗口标题栏的 title 属性改变为“Hello world”，如下图所示：



点击设计区右方的 show preview 图标，可以看到当前设置好的窗口，可见所做设置起了作用。如下图：



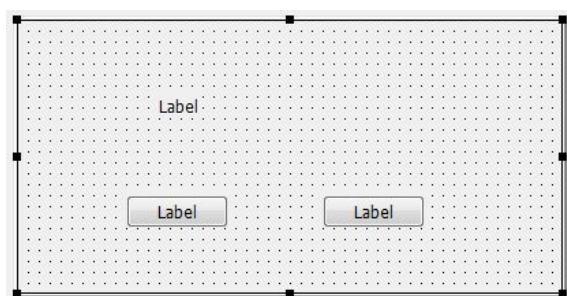
在设计区下方，有一排小按钮，用于向窗口中插入“控件”，如下图，用鼠标在其上滑过，可以看到其名称，如 wxBitmapButton、wxPanel、wxStaticText、wxButton 等。



可见，wxSmith 提供了相当丰富的控件。



下面，向你的窗口中加入四个控件，分别是：1 个 wxPanel，1 个 wxStaticText，2 个 wxButton。加入时，在控件栏中点击控件，然后，到窗口中的合适位置再点一下，控件即被加入。调整其位置，使看起来如下图所示：



逐个选择加入的控件，在“属性编辑区”观察并改变其中的部分属性：

- 静态文本框：其 Identifier 属性是 ID\_STATICTEXT1，将其 Label 属性中的值删除。
- 按钮 1：其 Identifier 属性是 ID\_BUTTON1，将其 Label 属性中的值改为&SayHello。
- 按钮 2：其 Identifier 属性是 ID\_BUTTON2，将其 Label 属性中的值改为&Quit。

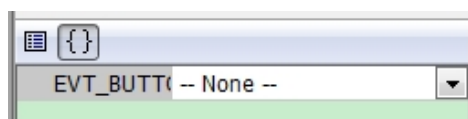
再点击设计区右方的 show preview 图标，观察设计好的窗口，如下所示：



## 5.2 为控件加入事件处理程序

应用程序中，需要为按钮设置 OnClick 事件，决定事件发生时要做的“动作”，也即要定义事件响应函数。

做法是，选中 Quit 按钮，在“属性编辑区”点击靠右边的 `{ }` 按钮设置“事件”。下图是初时的状态，--None--标示着现在还没有事件代码：



点击右边的向下三角，选择“--Add new handler--”，系统自动定义了一个函数体为空的函数：

```
void PlainFrame::OnButton2Click(wxCommandEvent& event)
{
}

```

现在需要做的，是加入退出程序需要的语句。加入后函数是：

```
void PlainFrame::OnButton2Click(wxCommandEvent& event)
{
    Close(true);
}

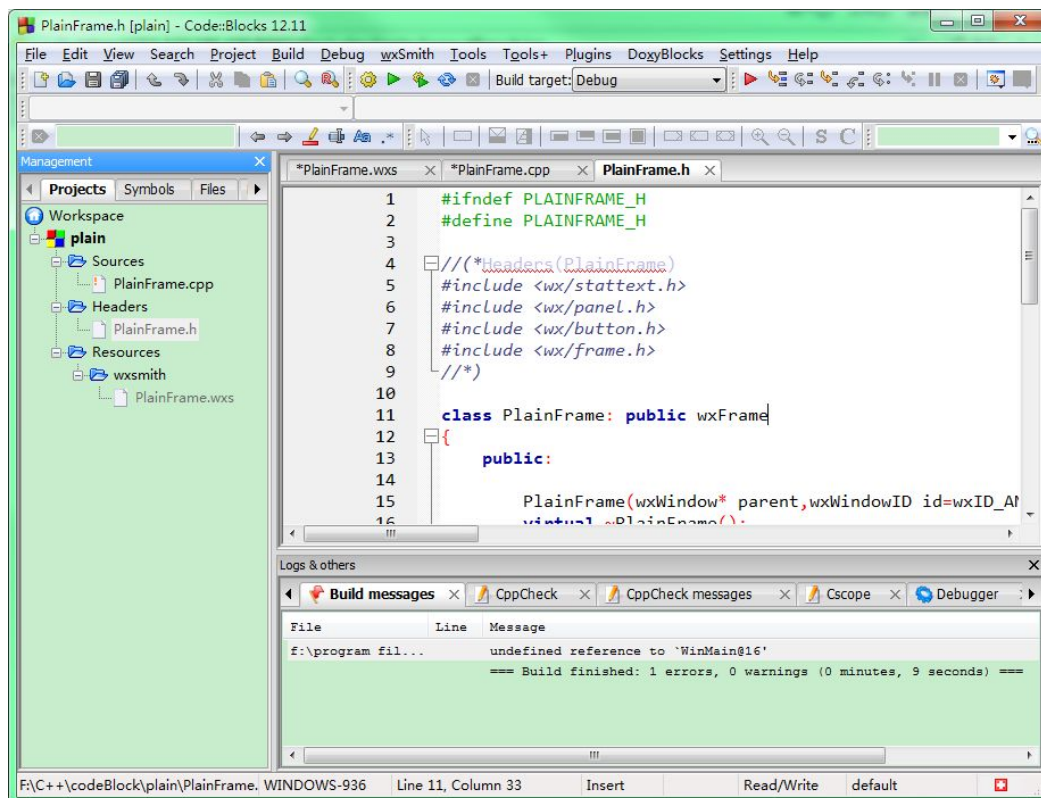
```

同样的方法，为 SayHello 按钮加上事件的处理，函数为：

```
void PlainFrame::OnButton1Click(wxCommandEvent& event)
{
    if(StaticText1->GetLabel().Contains(wxT("world")))
        StaticText1->SetLabel(StaticText1->GetLabel() + wxT(" again"));
    else
        StaticText1->SetLabel(wxT("Hello world"));
}

```

在整个应用程序的开发工作即将结束之际，我们可以完整观察一下已经开发出的程序。



在 Code::Blocks 左侧的 Management 窗口中，选择 Projects 页签，打开项目的 Sources，即源程序部分的 PlainFrame.cpp，以及 Headers 部分的 PlainFrame.h，可以看到的程序的全貌，如上图所示。

原来，前面利用 wxSmith 的所有设计，其结果都转变为了在前几章在看的代码。

wxSmith 利用可视化的界面，是在为我们自动生成程序！

这时候编译程序，如果有错误，参考 3.1.2 小节的处理方式，修改程序或配置环境。

最后，应该只有一个错误：undefined reference to `WinMain@16'。

注意到此时的程序，只是定义了窗口类，还缺少整个程序的主控部分。仿照在《wxWidgets tutorial》中见过的无数例子，在项目中加入下面的两个文件：

#### main.h

```
#include <wx/wx.h>

class MyApp : public wxApp
{
public:
    virtual bool OnInit();
};
```

#### main.cpp

```
#include "main.h"
#include "PlainFrame.h"

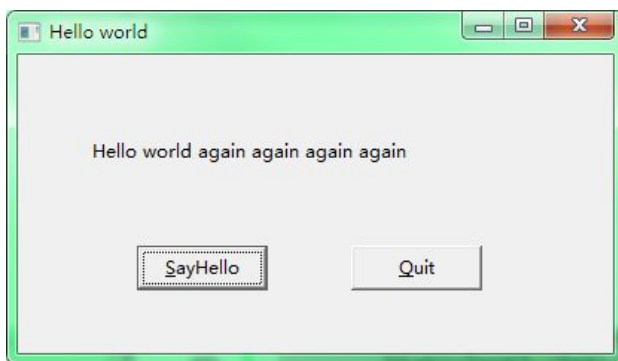
IMPLEMENT_APP(MyApp)

bool MyApp::OnInit()
{
    PlainFrame *plainFrame = new PlainFrame(NULL);
    plainFrame->Show(true);

    return true;
}
```

此时，可以让让运行程序了。下面是程序运行期间的一个截图：





## 5.3 写代码与拖控件

用 wxSmith 可以帮助程序员用可见的方式快速地做出应用程序的界面。一个问题是：p 这种“自动化”的手段，是不是让第 4 章辛苦看书手写代码所花的功夫白费了？

这个观点有些道理。有些业内人士描述他们的工作，就是拖控件。这种话，C++程序员说得少一些。用的开发工具越高级，越要这样说。

拖控件，也有水平高低，不必妄自菲薄。而确实，系统开发能力和水平，更多是在拖控件的背后。

第 4 章辛苦看书手写代码，是掌握 wxWidgets 构架的必要，是探知其根源的途径。优秀程序员要上得了厅堂（直接写代码），也下得了厨房（拖控件），哪样方便哪样来。以至于，拖控件解决不了的，就直接写代码完成。

能拖控件的人多，能写代码的人相对少。看第 5 章，不必等到第 4 章的功夫完全具备再说，而读完、练完了第 5 章，第 4 章一定要回头再好好看。

有了写代码能力的支撑，拖得一手好控件也有了保证。前面的 Hello world 只是一个开端，还有很多复杂的，也是构成良好交互功能的控件等着你拖好、用好。

## 5.4 深入学习的建议

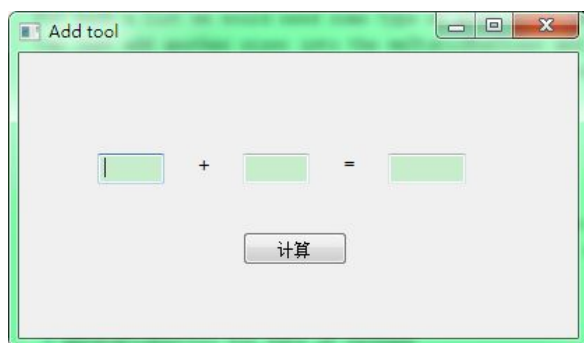
在 [http://wiki.codeblocks.org/index.php?title=WxSmith\\_tutorials](http://wiki.codeblocks.org/index.php?title=WxSmith_tutorials) 中提供了 9 个教程，英文版的。跟着做下去，会明白更多。用中学英语，很划算的一件事情。

附件中的“WxSmith\_tutorials.pdf”文档是这 9 个教程早期的一个版本，可以离线读这个教程。

可以自拟些小题目开发一下，逐渐就能过渡到开发项目。

势砖引玉，给几个小题目：

(1) 设计 GUI 程序，计算  $a+b=?$ ，参考界面如下，在前两个文本框中输入  $a$  和  $b$ ，点击“计算”按钮后，在第 3 个文本框中出  $a+b$  的结果。



(2) 用三个文本框输入一元二次方程  $ax^2 + bx + c = 0$  的三个系数，解出方程的根。可以定义一个一元二次方程类来解决有关的计算问题。

(3) 不再写了，计算器、记事本、画图板，适合的题目，多的是了。学会拟小题目，这也是课外自主学习能找到的感觉。

## 附：学习材料清单

- wxWidgets 的官网: <http://www.wxwidgets.org/downloads/>
- wxWidgets 的下载页面: <http://www.wxwidgets.org/downloads/>
- 在线教程《wxWidgets tutorial》: 网址在 <http://zetcode.com/gui/wxwidgets/>, 本文提供了一个整理后的 word 版本 (即**附件 1**)
- 《使用 wxWidgets 进行跨平台程序开发》:  
<http://www.amazon.cn/gp/product/B00A1WDQ30>, 部分电子版 (含书中例程的源代码)  
链接: <http://download.csdn.net/detail/cjylg/2997827>, 该书原版《Cross Platform GUI Programming With wxWidget》: <http://www.wxwidgets.org/docs/book/>
- wxwidgets 的 Wiki 主页: [http://wiki.wxwidgets.org/Main\\_Page](http://wiki.wxwidgets.org/Main_Page)
- Wxwidgets Wiki 的 Guides & Tutorials: [http://wiki.wxwidgets.org/Guides\\_%26\\_Tutorials](http://wiki.wxwidgets.org/Guides_%26_Tutorials)
- 深入 wxSmith 的教程: [http://wiki.codeblocks.org/index.php?title=WxSmith\\_tutorials](http://wiki.codeblocks.org/index.php?title=WxSmith_tutorials)
- 深入 wxSmith 的教程 (老版本) pdf 版: 见**附件 2**